University of Alberta

SUBSPACE CLUSTERING METHODS FOR HIGH DIMENSIONAL DATA

by

Gabriela Moise ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2008

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-46390-1
Our file Notre référence
ISBN: 978-0-494-46390-1

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

University of Alberta

Library Release Form

Name of Author: Gabriela Moise

Title of Thesis: Subspace Clustering Methods for High Dimensional Data

Degree: Doctor of Philosophy

Year this Degree Granted: 2008

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Gabriela Moise

Date: ____________

Abstract

Prominent research has shown that increasing data dimensionality results in the loss of contrast in distances between data points. Thus, clustering algorithms measuring the similarity between data points based on all features/attributes of a data set tend to break down in high dimensional spaces. In addition, not all attributes of a data set may be relevant for the clustering analysis.

Motivated by these observations, it has been hypothesized that data points may form clusters only when a subset of the attributes, i.e., a *subspace*, is considered. Furthermore, data points may belong to different clusters in different subspaces.

Subspace and projected clustering techniques search for clusters of points in subsets of attributes. Subspace clustering enumerates clusters of points in all subsets of attributes, typically producing many overlapping clusters. Projected clustering computes several disjoint clusters, plus outliers, so that each cluster exists in its own subset of attributes.

In this thesis, we propose three novel techniques that advance the state-of-the-art in the subspace and projected clustering field. First, we propose a projected clustering technique P3C that 1) depends on parameters that can be set without prior knowledge about the data; 2) can effectively discover low dimensional clusters embedded in high dimensional spaces; 3) can compute disjoint or overlapping clusters. Second, we propose two extensions that make P3C the first projected clustering technique that can be applied on both numeri-

cal and categorical data sets. Third, we propose a novel problem formulation for subspace and projected clustering that aims at extracting non-redundant, axis-parallel, statistically significant regions from the data. The problem formulation is given as an optimization problem, for which exhaustive search is not a viable solution because of computational infeasibility. Therefore, we propose an approximation algorithm, STATPC, that has the same advantageous features as P3C, but, in addition, guarantees that its solution stands out in the data in a statistical sense, and it is not just an artefact of the method.

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Subspace Clustering Methods for High Dimensional Data** submitted by Gabriela Moise in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in *Database Systems.*

Jörg Sander

Marek Reformat

Howard Hamilton

Osmar Zaïane

Dale Schuurmans

Date: ____________________

To my husband, Daniel.
Without you, I wouldn't have succeeded.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Jörg Sander, for encouragement, support, insightful discussions, and constant feedback.

I would like to thank the members of my examining committee for their time and effort in reading my candidacy proposal and dissertation and attending the candidacy and defence examinations.

The research supporting this dissertation also greatly benefited from collaborations with Dr. Martin Ester from Simon Fraser University and Dr. Peer Kröger and Dr. Arthur Zimek from Ludwig-Maximilians University of München.

I would also like to thank the authors who provided us with the implementation of some of the algorithms relevant for the subject of this thesis.

I would like to acknowledge the financial support from Alberta Ingenuity Fund and the iCORE Circle of Research Excellence.

Table of Contents

1 **Introduction** **1**
1.1 Overview of Related Work 3
1.2 Contributions 5
1.3 Outline 6

2 **Related Work** **8**
2.1 Projected Clustering Techniques 8
2.1.1 Partitional Approaches 9
2.1.2 Hierarchical Approaches 13
2.1.3 Density-based Approaches 14
2.2 Subspace Clustering Techniques 18
2.3 Categorical Subspace and Projected Clustering 22
2.4 Related Problem Formulations 24

3 **P3C: Projected Clustering via Cluster Cores** **28**
3.1 Preliminary Definitions 29
3.2 Overview of P3C 30
3.3 Approximating Projections of Characteristic Hyper-rectangles 32
3.4 Cluster Cores Computation 33
3.5 Cluster Cores Refinement, Outlier Detection, Relevant Attributes Refinement 35
3.6 A Note on Parameters 37
3.7 Theoretical Complexity 38
3.8 Experimental Evaluation 38
3.8.1 Compared Techniques 39
3.8.2 Synthetic Data 39
3.8.3 Real Data 40
3.8.4 Experimental Setup 40
3.8.5 Performance Measures 41
3.8.6 Accuracy Results 41
3.8.7 Robustness to Noise 46
3.8.8 Sensitivity Analysis 47
3.8.9 Scalability Experiments 47
3.9 Summary 49

4 **P3C for Categorical Data** **51**
4.1 Preliminary Definitions 51
4.2 Overview of P3C for Categorical Data 52
4.3 Approximating Projections of Characteristic Hyper-rectangles on Categorical Attributes 52
4.4 Cluster Cores Refinement and Outlier Detection on Categorical Data 55
4.5 A Note on Parameters 56

4.6 Theoretical Complexity 56
4.7 Experimental Evaluation 57
  4.7.1 Compared Techniques 57
  4.7.2 Synthetic Data 57
  4.7.3 Real Data 58
  4.7.4 Experimental Setup 58
  4.7.5 Performance Measures 58
  4.7.6 Accuracy Results 58
  4.7.7 Robustness to Noise 63
  4.7.8 Sensitivity Analysis 63
  4.7.9 Scalability Experiments 64
4.8 Summary 65

5 Finding Non-Redundant, Statistically Significant Regions in High Dimensional Data 68
5.1 Overview 69
5.2 Statistical Significance 69
5.3 Relevant vs. Irrelevant Attributes 70
5.4 Redundancy-oblivious Problem Definition 73
5.5 Explain Relationship 75
5.6 Redundancy-aware Problem Definition 82

6 Approximation Algorithm STATPC 85
6.1 Finding True Subspace Clusters Around Data Points 85
  6.1.1 Detecting Candidate Subspaces 87
  6.1.2 Detecting a Locally Optimal Subspace Cluster per Candidate Subspace 93
  6.1.3 Detecting a Locally Optimal Subspace Cluster Between Locally Optimal Subspace Clusters in Candidate Subspaces 95
  6.1.4 Constructing the Set $R^{reduced}$ 96
6.2 Greedy Optimization 96
6.3 A Note on Parameters 97
6.4 Theoretical Complexity 98
6.5 Experimental Evaluation 98
  6.5.1 Compared Techniques 98
  6.5.2 Synthetic Data 98
  6.5.3 Real Data 100
  6.5.4 Experimental Setup 100
  6.5.5 Performance Measures 101
  6.5.6 Statistical Significance of Results 101
  6.5.7 Accuracy Results 101
  6.5.8 Sensitivity Analysis 117
  6.5.9 Scalability Experiments 119
6.6 Summary 121

7 Conclusions and Future Work 123
7.1 Directions for Future Work 124

Bibliography 126

A Statistical Hypothesis Testing 131

List of Tables

6.1 Analysis of different criteria in data generation 112
6.2 Analysis of different criteria in data generation (cont.) 113

List of Figures

3.1 Two 2D projected clusters 29
3.2 Pseudo-code of P3C 31
3.3 P3C and competing techniques on category Uniform_Equal . . 42
3.4 P3C and competing techniques on category Uniform_Different 43
3.5 P3C and competing techniques on category Gaussian_Equal . 44
3.6 P3C and competing techniques on category Gaussian_Different 45
3.7 Accuracy of P3C and competing techniques on E.coli data set 45
3.8 Accuracy of P3C and competing techniques on Glass data set 46
3.9 Accuracy of P3C and competing techniques on Iris data set . . 46
3.10 Robustness of P3C and competing techniques to noise 47
3.11 Sensitivity of P3C to parameter α_{Binom} 48
3.12 Scalability of P3C and competing techniques with increasing database size 48
3.13 Scalability of P3C and competing techniques with increasing database dimensionality 49
3.14 Scalability of P3C and competing techniques with increasing average cluster dimensionality 49

4.1 Illustration of intervals on categorical attributes 54
4.2 P3C and competing techniques on categorical category Equal . 59
4.3 P3C and competing techniques on categorical category Different 60
4.4 Accuracy of P3C and competing techniques on Congressional Votes data set 61
4.5 Accuracy of P3C and competing techniques on Post-Operative Patient data set 61
4.6 Accuracy of P3C and competing techniques on Hepatitis data set 61
4.7 Accuracy of P3C and competing techniques on Contraceptive Method Choice data set 62
4.8 Accuracy of P3C and competing techniques on Flags data set 62
4.9 Robustness of P3C and competing techniques to noise on categorical data 63
4.10 Categorical data: sensitivity of P3C to parameter α_{Binom} . . . 64
4.11 Categorical data: scalability of P3C and competing techniques with increasing database size 64
4.12 Categorical data: scalability of P3C and competing techniques with increasing database dimensionality 65
4.13 Categorical data: scalability of P3C and competing techniques with increasing average cluster dimensionality 66
4.14 Categorical data: scalability of P3C and competing techniques with increasing domain size per attribute 66

5.1 (a)-(f) Redundancy in R (solid/dotted lines for true/"induced" subspace clusters) (g) Example data 74
5.2 Illustration of $\{A\}$ explains B, $\{B\}$ does not explain A 78

5.3 Illustration of $\{A\}$ explains B, $\{B\}$ explains C, $\{A\}$ does not explain C 80
5.4 The existence of subspace clusters B and C is due to the existence of embedded subspace cluster A 84

6.1 Pseudo-code of STATPC 86
6.2 The issue of committing to a candidate subspace 91
6.3 Pseudo-code of detecting candidate subspaces around a point Q 94
6.4 Pseudo-code of detecting a locally optimal subspace cluster around a data point Q in a candidate subspace S. 96
6.5 Pseudo-code of detecting greedily a solution P^{sol} on $R^{reduced}$. . 97
6.6 STATPC and competing techniques on category Uniform_Equal 102
6.7 STATPC and competing techniques on category Uniform_Different 103
6.8 STATPC and competing techniques on category Gaussian_Equal 104
6.9 STATPC and competing techniques on category Gaussian_Different 105
6.10 The effect of data dimensionality d on STATPC and competing techniques 106
6.11 The effect of database size n on STATPC and competing techniques 107
6.12 The effect of the number of clusters k on STATPC and competing techniques 108
6.13 The effect of cluster sizes on STATPC and competing techniques 109
6.14 The effect of extent of clusters in relevant attributes on STATPC and competing techniques 110
6.15 The effect of overlap of clusters in common relevant attributes on STATPC and competing techniques 111
6.16 Accuracy of STATPC and competing techniques on Pima Indians Diabetes data set 114
6.17 Accuracy of STATPC and competing techniques on Liver Disorders data set 114
6.18 Accuracy of STATPC and competing techniques on WPBC data set 114
6.19 Accuracy of STATPC and competing techniques on Glass data set 115
6.20 Accuracy of STATPC and competing techniques on Iris data set 116
6.21 Sensitivity of STATPC to parameter α_0 117
6.22 Sensitivity of STATPC to parameter α_K 118
6.23 Sensitivity of STATPC to parameter α_H 118
6.24 Scalability of STATPC and competing techniques with increasing database size 119
6.25 Scalability of STATPC and competing techniques with increasing database dimensionality 120
6.26 Scalability of STATPC and competing techniques with increasing average cluster dimensionality 120

A.1 Two-sided statistical hypothesis test 132

Chapter 1

Introduction

Cluster analysis is defined as the process of organizing a set of data objects into groups or *clusters* so that objects within a group are more similar to each other than to objects from other groups. Typically, the objects to be clustered are represented by points in a multi-dimensional feature space, and a distance function is used to measure the dissimilarity between the corresponding multi-dimensional points.

Cluster analysis has been extensively studied in many areas, including the database, machine learning and statistics communities. It is often used as a stand-alone tool to gain insight into the data distribution, or as a preliminary step for subsequent analyzes. Clustering has been applied to a large number of practical problems, such as market segmentation, spatial data analysis, gene expression data analysis, etc.

Numerous clustering algorithms have been proposed in the literature. They are often classified into *partitional* (e.g., KMeans [49], PAM [45]), *hierarchical* (e.g., AGNES, DIANA [45]), *density-based* (e.g., DBSCAN [27], OPTICS [10], DENCLUE [41]), *grid-based* (e.g., STING [70], WaveCluster [64]), *spectral* (e.g., [59]), and *model-based* (e.g., EM [25]) techniques. Comprehensive surveys of these clustering techniques and concepts can be found in [42], [40], [16], [75].

Significant progress has been made during the last decade towards making clustering algorithms 1) scalable to large data sets (e.g., CLARA [45], CLARANS [56], BIRCH [85], Data Bubbles [20]), 2) robust to noise and capable of discovering clusters of various shapes and densities (e.g., DBSCAN [27], OPTICS [10], CURE [38], CHAMELEON [44]), 3) non-parametric or at least robust to the parameters required (e.g., TURN [29], WaveCluster [64]), 4) suitable for spatial data (e.g. [83], [71], [72]), or 5) incorporate available domain knowledge (e.g., [13], [73]).

Traditionally, clustering algorithms measure the similarity between data points by considering all features/attributes of a data set. These approaches are successful for low dimensional data sets. However, in high dimensional data sets, these clustering algorithms tend to break down both in terms of accuracy, as well as efficiency, due to a lack of contrast in distances between data points. Seminal research [17] has shown that, as the dimensionality increases, the farthest neighbor of a data point is expected to be almost as close as its nearest neighbor for a wide range of data distributions and distance functions. Due to this effect, the concept of proximity, and subsequently the concept of a "cluster", are seriously challenged in high dimensional spaces.

At the same time, automatic data collection facilities have become increasingly available, and thus, an increasing number of features/attributes can be automatically measured. However, not all of these attributes may be relevant for the clustering analysis. The irrelevant attributes may in fact "hide" the clusters by making two data points that belong to the same cluster look as dissimilar as an arbitrary pair of data points.

Motivated by these observations, it has been hypothesized [7] that data points may form clusters only when a subset of the attributes, i.e., a *subspace*, is considered. Furthermore, data points may belong to different clusters in different subspaces.

As a motivating example, let us consider a gene expression data set that measures the expression level of human genes in several human tissues. When clustering tissues, we deal with a clustering problem in a very high dimensional space, because the number of genes, typically in the thousands, is several orders of magnitude larger than the number of tissues, usually in the tens. Because of the sparse nature of the data, it is unlikely that data points, representing tissues, form clusters in full dimensional space. Instead, data points may form clusters only when a small number of "relevant" attributes are considered. As noted in the bio-medical literature, only a relatively small number of genes out of the total number of genes may be relevant for distinguishing between normal and cancerous tissues. Furthermore, data points may form different clusters in different subsets of attributes, depending on the different phenotypes represented by these attributes. For example, the cancerous tissues may form a cluster when a certain subset of attributes is selected, whereas the normal tissues may form a cluster when a different subset of attributes is selected. The selected attributes are potential indicators for the presence, respectively absence, of cancer.

The *subspace clustering problem* is the task of automatically determining clusters of points in different, possibly overlapping, subspaces of a data set.

Global dimensionality reduction techniques such as feature selection and feature transformation (e.g., Principal Component Analysis (PCA)) are not effective for the subspace clustering problem. These techniques cluster data only in a particular subspace, in which it may not be possible to recover all clusters, and information concerning points clustered differently in other subspaces is lost [57].

1.1 Overview of Related Work

In this thesis, we focus on the problem formulation in which a subspace is defined as a subset of the original attributes of a data set. In this case, the discovered subspace clusters are easily interpretable by the user because the original attributes have meaning in real-life applications. A related problem formulation is one in which a subspace is defined as an arbitrary set of orthogonal vectors [6].

Techniques for discovering clusters of points in subsets of attributes have been classified into two categories [57]: *subspace clustering* techniques, and *projected clustering* techniques. Both types of techniques are similar in the sense that they discover clusters of points that exist in subspaces of a data set. From this point of view, both types of techniques are "subspace clustering" techniques [1]. However, they differ in their problem definition, and in their strengths and weaknesses.

Subspace clustering techniques search for all clusters of points in *all* subspaces of a data set according to their respective cluster definition. Existing subspace clustering techniques start with one-dimensional clusters, which are subsequently merged bottom-up, similarly to the Apriori algorithm for finding frequent itemsets [8], in order to compute clusters of higher dimensionality. To avoid an exhaustive search through all possible subspaces, the cluster definition is based on a global density threshold that ensures anti-monotonic properties necessary for an Apriori style search. However, the cluster definition ignores the fact that density decreases with dimensionality. Large values for the global density threshold will result in only low dimensional clusters, whereas small values for the global density threshold will result in a large number of low dimensional clusters (many of which are meaningless), in addition to the higher dimensional clusters. Some subspace clustering techniques use an axis-aligned grid for estimating the density of a region in the data space. These techniques are sensitive to the resolution of the grid used, and they may miss clusters that are inadequately oriented or shaped with respect to the grid positioning.

Projected clustering techniques define a projected cluster as a pair (X, Y), where X is a subset of data points, and Y is a subset of data attributes, called the "relevant" attributes, so that the points in X are "close" when projected on each of the attributes in Y, but they are "not close" when projected on each of the remaining attributes, called the "irrelevant" attributes. Projected clustering techniques have an explicit or implicit measure of "closeness" on relevant attributes (e.g., small range/variance), and a "non-closeness" measure on irrelevant attributes (e.g., uniform distribution/large variance). A search method will report all projected clusters in the particular search space that a technique considers. If only k projected clusters are desired, the techniques typically use an objective function to define what the optimal set of k projected clusters is.

---

[1] We believe that density-based subspace clustering would be a more appropriate name than subspace clustering for the techniques in this category. However, in order to preserve the terminology used in literature, we will use in this thesis the term of subspace clustering techniques.

Existing projected clustering techniques are either based on the computation of k initial clusters in full dimensional space, or leverage the idea that clusters with as many relevant attributes as possible are preferable. Consequently, these techniques are likely to be less effective in the practically most interesting case of projected clusters with very few relevant attributes, because the members of such clusters are likely to have low similarity in full dimensional space.

In addition, a re-occurring weakness of both these types of techniques is that their performance depends greatly on a series of parameters whose appropriate values are difficult to anticipate by the users (e.g., the number of projected clusters or the average dimensionality of subspaces where clusters exist).

From an algorithmic point of view, all subspace clustering techniques are based on an Apriori-like, bottom-up discovery of clusters based on some global density thresholds. They typically report a large number of overlapping clusters. There is much more diversity in the existing projected clustering techniques that can be classified, just like the full-dimensional algorithms, into partitional, hierarchical and density-based techniques. The majority of projected clustering techniques compute disjoint clusters; others have the option to assign data points to more than one cluster.

The majority of existing subspace and projected clustering techniques are designed for *numerical* data sets, i.e., data sets where the domain of every attribute is inherently ordered. However, many real data sets are *categorical*, i.e., the attribute domains are discrete and not ordered.

Subspace and projected clustering techniques designed for numerical data are not readily applicable to categorical data sets. Some of these techniques use distance functions that exploit the geometric properties of the data space, which cannot be effectively captured by categorical distance functions, such as the matching coefficient. Other techniques require numerical computations that are not well-defined for categorical attributes (e.g., mean, variance, eigenvalues). Finally, some techniques are based on the discretization of individual data attributes into bins, and the notion of "neighboring" bins is used to manage the search through all possible subspaces. Categorical attributes lack order, and thus this notion is not directly applicable.

A significantly smaller body of work has been dedicated to the subspace clustering problem on categorical data than to the same problem on numerical data. Existing subspace and projected clustering techniques for categorical data exhibit the same weaknesses as their numerical counterparts.

1.2 Contributions

In this thesis, we propose three novel techniques that advance the state-of-the-art in the subspace and projected clustering field.

First, we propose a projected clustering technique, called **P3C** (**P**rojected **C**lustering via **C**luster Cores) with the following properties:

1. P3C can effectively discover projected clusters in the data while requiring parameters that represent the error probability that the user is willing to accept. Therefore, setting these parameters requires no prior knowledge about the data, and, in contrast to most previous approaches, there is no need to provide the target number of clusters as input.
2. P3C can effectively discover low dimensional clusters embedded in high dimensional spaces.
3. P3C may assign a data point to more than one cluster if the data point satisfies the description of more than one cluster.
4. P3C is robust with respect to noise.

P3C is comprised of several steps. First, regions corresponding to projections of clusters on individual attributes are computed. Second, *cluster cores* are identified by spatial areas that 1) correspond to a combination of the detected regions, and 2) contain an unexpectedly large number of points. Third, cluster cores are refined into projected clusters, outliers are identified, and relevant attributes for each cluster are refined.

Second, we propose to extend P3C for categorical data. We propose two adaptations that need to be performed in order to make P3C applicable on categorical data:

1. We adapt the computation of cluster projections on individual attributes for categorical attributes.
2. We adapt the refinement of cluster cores into projected clusters and the computation of outliers for categorical data.

P3C is the first projected clustering technique that can be applied on both numerical and categorical data sets. P3C for categorical data exhibits the same properties as P3C for numerical data.

Third, we observe that one problem common to many existing subspace and projected clustering techniques is that their objectives are stated in a way that is not independent of the particular algorithm that is proposed to detect such clusters in the data. A second problem is the definition of cluster density based on user-defined parameters, which makes it hard to assess whether the reported clusters are an artefact of the algorithm or whether they actually stand out in the data in a statistical sense.

Motivated by these observations, we propose a novel problem formulation that aims at extracting axis-parallel regions that stand out in the data in a statistical sense. The set R of all axis-parallel, statistically significant regions that exist in a data set is typically highly redundant. Therefore, we propose to represent the set R through a reduced set of axis-parallel, statistically significant regions that in a statistically meaningful sense *explains* the existence of all the regions in R. We formalize these notions and we formulate the task of representing R through a reduced set of "explaining" regions as an optimization problem.

Exhaustive search is not a viable solution for solving the optimization problem because of computational infeasibility. Therefore, we propose an approximation algorithm, called **STATPC**, with the following properties:

1. The solution computed by STATPC stands out in the data in a statistical sense, and it is not just an artefact of the method.
2. The parameters required by STATPC are error probabilities that the user is willing to accept, and thus, setting these parameters does not require prior knowledge about the data.
3. STATPC can effectively discover clusters in the data, even when these clusters have low dimensionality with respect to the total dimensionality of the data set.
4. STATPC can assign a data point to more than one cluster.
5. In a comprehensive experimental evaluation, we study the performance of STATPC over a variety of parameters involved in the data generation model. We show that STATPC significantly outperforms existing subspace and projected clustering techniques in terms of accuracy. In addition, we discuss the strengths and weaknesses of the compared techniques. Our results can be used as a guide for the data mining practitioner to select which techniques are preferable in certain scenarios.

The results of our research have been published in [52], [53], and [51].

1.3 Outline

The rest of the thesis is organized as follows. Chapter 2 surveys the work relevant for this thesis. Chapter 3 describes the technique P3C for numerical data. Chapter 4 describes P3C for categorical data. The novel problem formulation for subspace and projected clustering that we propose is presented in chapter 5. The approximation algorithm STATPC for the novel problem definition is described in chapter 6. Conclusions and directions for future work are given in chapter 7. Appendix A summarizes the methodology of statistical hypothesis testing.

Chapter 2

Related Work

This chapter surveys the work relevant for this thesis. Sections 2.1 and 2.2 discuss existing projected and subspace clustering techniques. Projected and subspace clustering techniques for categorical data are surveyed in Section 2.3. An overview of related problem formulations is given in Section 2.4.

2.1 Projected Clustering Techniques

Projected clustering techniques define a projected cluster as a pair (X, Y), where X is a subset of data points and Y is a subset of attributes so that 1) the data points in X project along each attribute $a \in Y$ on a range of values that is "small" compared to the range of values on which the whole data set projects on a, and 2) the data points in X project along each attribute a' *not* in Y on a range of values that is "comparable" to the range of values on which the whole data set projects on a'. For a projected cluster (X, Y), the attributes in Y are called the "relevant" attributes for X, whereas the remaining attributes are called "irrelevant" for X. Projected clustering techniques have, implicitly or explicitly, some notions of "small" and "comparable" in the definition of a projected cluster.

The data model in projected clustering assumes that a data set D consists of K projected clusters, $\{(X_i, Y_i)\}_{i\in\{1,\ldots,K\}}$, and a set of outliers, O, where $\{X_1, \ldots, X_K, O\}$ form a partition of D. The subsets of attributes $\{Y_i\}_{i\in\{1,\ldots,K\}}$ may not be disjoint and they may have different cardinalities. The outliers O are assumed to be uniformly distributed throughout the data space. The projected clustering problem is to detect K projected clusters in the data, plus possibly a set of outliers.

Projected clustering techniques proposed in the literature can be classified into partitional (Section 2.1.1), hierarchical (Section 2.1.2), and density-based (Section 2.1.3) techniques.

Partitional projected clustering techniques use an objective function to define what is the optimal set of K projected clusters. These techniques address the optimization problem in an iterative manner: first, K "tentative" clusters are computed in full dimensional space; second, sets of relevant attributes for each tentative cluster are computed; and, finally, the tentative clusters are refined based on the relevant attributes just computed. This iterative process is repeated several times, and the solution corresponding to the best objective function value is kept. These techniques report disjoint clusters, and they use some heuristics to identify outliers.

Hierarchical projected clustering techniques are guided in their computation of clusters by the idea that clusters with many relevant attributes are preferable to clusters with few relevant attributes. Like the partitional projected clustering techniques, these techniques require the desired number of clusters K as a parameter, compute disjoint clusters, and use some heuristics to identify outliers.

Density-based projected clustering techniques assume that projected clusters have a certain density. There are several flavors of techniques in this category, as discussed in Section 2.1.3. These techniques do not require the desired number of clusters as a parameter, and some of them are able to compute overlapping clusters, i.e., clusters that share data points. However, these techniques require various other parameters.

Typically, projected clustering techniques require parameters that are difficult to set by users (e.g., the number of projected clusters or the average number of relevant attributes of projected clusters), and they are sensitive to the values of these parameters.

Moreover, projected clustering techniques are less effective for discovering projected clusters with few relevant attributes embedded in high dimensional spaces, because these techniques are either based on the computation of tentative clusters in full dimensional space - which often do not represent well the low dimensional projected clusters in the data - or they tend to prefer clusters with many relevant attributes.

Finally, many projected clustering techniques restrict the membership of a data point to at most one projected cluster. Although this effect may be desirable in some applications, it is preferable to have techniques that leave to the user the decision whether the computed clusters should be disjoint or not.

In the following subsections, we summarize projected clustering techniques proposed in the literature, and we discuss their potential drawbacks.

2.1.1 Partitional Approaches

PROCLUS [5] represents each cluster by one of its points, called a "medoid", together with its set of relevant attributes. PROCLUS minimizes the average within-cluster dispersion, which is defined as the average Manhattan *segmental* distance [1] between the members of a cluster and the cluster medoid.

---

[1] The Manhattan segmental distance between a data point and a medoid is their Manhattan distance computed in the relevant subspace of the medoid, and normalized by the dimensionality of this subspace.

PROCLUS requires two critical input parameters: K, the desired number of clusters, and l, the average cluster dimensionality. It consists of three phases: an initialization, an iterative and a refinement phase.

In the initialization phase, the technique randomly samples $A * K$ data points, and from this sample, it greedily selects a set M of $B * K$ scattered medoids, with the goal of selecting at least one medoid from each cluster. A and B are user-defined parameters.

In the iterative phase, PROCLUS first selects K arbitrary medoids from the set M. Second, for each of the K medoids, it computes a cluster, as explained shortly. The average within-cluster dispersion of the current clustering solution is computed, and the clustering solution is recorded if it is the solution with the minimum dispersion obtained so far. PROCLUS tries to improve iteratively the current solution by detecting a "bad" medoid, replacing it with a random medoid from M, and re-computing clusters around the medoids. A medoid is "bad" if its cluster has few points. If the current solution cannot be improved after a certain number of replacements have been tried, then the technique terminates, and this solution is fed into the refinement phase.

Given a medoid, PROCLUS computes its cluster in three steps, as follows. First, it computes a tentative cluster around the medoid by selecting the data points that are within a certain distance from the medoid in full dimensional space. Second, for each tentative cluster, its relevant attributes are determined as attributes where the average distance between the points in the tentative cluster and the medoid is "small" compared with the same average distance computed for the other attributes. Note that the objective function of PROCLUS cannot be directly minimized for determining the relevant attributes, because the less relevant attributes a cluster has, the lower the average-within cluster dispersion. Therefore, a user-specified average dimensionality l is introduced in order to decide how many attributes with "small" average distances should be selected for a cluster. Third, data points are assigned to the closest medoid in terms of Manhattan segmental distance.

In the refinement phase, the clustering solution obtained at the end of the iterative phase is used to re-compute relevant attributes for each cluster. Subsequently, a data point is assigned to the closest medoid, unless there is another medoid closer to this medoid than the data point, in which case the point is declared an outlier (closeness is measured by Manhattan segmental distances).

PROCLUS tends to compute clusters that are hyper-spherical in shape, and which exist in subspaces of approximately equal dimensionality. Due to the sampling step, PROCLUS may miss clusters with a small number of points. The performance of PROCLUS crucially depends on the two required input parameters K and l, whose appropriate values are difficult to guess. Another weakness is the strong dependency on the initial clustering which is hard to determine since it is performed in the full-dimensional space where the "true" distances will be distorted by noisy attributes.

FINDIT [74] starts by selecting randomly two sets: S, a reduced version of the data set, and M, a set of medoids. The sizes of S and M are determined using Chernoff bounds so that any cluster with more than $C_{minsize}$ points has, with high probability, at least a certain number of points in S, and at least 1 point in M. $C_{minsize}$ is a user-defined parameter.

For each medoid in M, a tentative cluster is computed by selecting its V nearest neighbors from S. FINDIT measures the distance between two data points with the "dimension-oriented-distance" (*dod*), which represents the number of attributes in which the points are farther than a given ϵ. The members of a tentative cluster are points that are within distance ϵ from the medoid on as many attributes as possible. V is a user-defined parameter that should be larger than $C_{minsize}$.

Tentative clusters are used to determine relevant attributes for the clusters. An attribute is considered relevant for a cluster if, on this attribute, a certain percentage of the tentative cluster's members are within ϵ distance from the cluster's medoid.

Since there is a set of relevant attributes for each medoid in M, the tentative clusters can be refined. A data point is assigned to a medoid if the data point is within ϵ distance from the medoid on all medoid's relevant attributes.

Finally, the clusters formed around medoids in M are clustered using agglomerative hierarchical clustering. The distance used is the *dod* distance between two medoid clusters. The hierarchical algorithm ends when the distance between pairs of medoid clusters is greater than a user-specified parameter $D_{mindist}$. The resulting medoid clusters are refined by removing those that are too small or by merging some of them.

The ϵ parameter controls the "resolution" at which clustering is performed; thus several values for ϵ are tried, and the best solution according to a quality measure based on size and dimensionality of clusters is reported.

FINDIT is sensitive to the numerous parameters used, and it has difficulties in finding low dimensional clusters. In addition, FINDIT has a large running time due to the multiple values for ϵ tried.

SSPC [80] is similar in structure to PROCLUS, and it uses an objective function based on the relevance score of HARP [79] (described below in Section 2.1.2). An attribute a is relevant for a set of data points X if the variance of the projections of the points in X along attribute a is m times smaller than the variance of the projections of all data points along attribute a. m is a user-defined parameter.

SSPC starts by determining, for each cluster, a set of representative points and relevant attributes. If available, domain knowledge in the form of labeled data points and/or attributes is used to improve the quality of the representative points and their relevant attributes.

When no domain knowledge is available, the first representative point of the first cluster is selected at random. Attributes with high density around this representative point are considered relevant, and three-dimensional (3D) grids are built with these relevant attributes. The set of representative points for the first cluster consists of points that are located in cells with high density in the 3D grids that are neighbors of the cell where the first representative point is located. Subsequently, for each of the following clusters, the first representative point is selected such that its minimum distance to representative points of other clusters is maximized. The same procedure based on 3D grids is repeated to determine a set of representative points and relevant attributes for each of the subsequent clusters.

If some points of a cluster are known, these points can be used as representative points for the cluster, and also, can be used for identifying relevant attributes for the cluster, which are subsequently used in the 3D grids. If some relevant attributes of a cluster are known, these attributes can be included in the set of attributes that are used in the 3D grids.

In each clustering round, for each cluster, a representative point from the associated set of representative points is chosen, and each data point is assigned to the representative point that gives the greatest improvement in the objective function. If a data point does not improve the quality score of any cluster, it is put in the outlier list. Subsequently, the relevant attributes for each cluster are determined so that the objective function is maximized. The quality of the current clustering solution is recorded if it is the best solution obtained so far. The best solution is restored, a "bad" representative point (i.e., a representative point from a small cluster) is identified and replaced with another representative point, and the process is repeated until the current best solution has not changed for a user-defined number of consecutive iterations.

SSPC requires the number of clusters K as a parameter, and its performance depends on the selection threshold m used to determine relevance scores of attributes.

ORCLUS [6] is a technique designed for the more general problem of detecting clusters of points in arbitrary sets of orthogonal vectors, but it can be applied for the case when the sets of orthogonal vectors are the original data attributes.

ORCLUS partitions a data set into K clusters, plus possibly a set of outliers, where each cluster has the same number l of principal directions with low variance. K and l are user-defined parameters.

ORCLUS starts by randomly selecting k_0 data points as seeds. Initially, the dimensionality l equals the data dimensionality, d. ORCLUS progressively reduces the number of clusters from k_0 to K by a factor α, and, accordingly, the cluster dimensionality from d to l, by a factor β. At each step, ORCLUS computes k_c clusters of dimensionality l_c, by performing an *Assignment* step, a *Find Vectors* step, and a *Merge* step. In the *Assignment* step, data is partitioned into k_c current clusters by assigning each data point to the nearest seed, using the projected Euclidean distance in the subspace associated with the seed. Also, each seed is replaced by the centroid of the cluster. In the *Find Vectors* step, ORCLUS finds the subspace ϵ_i of dimensionality l_c for each current cluster C_i by (a) computing the covariance matrix of the cluster C_i, (b) selecting ϵ_i as being the l_c eigenvectors corresponding to the smallest l_c eigenvalues, and (c) reducing l_c by a factor β to l_{new}. In the *Merge* step, clusters with similar orientations, i.e., clusters having the least spread directions similar, should be merged. If C_i and C_j have similar least spread directions, then the projected energy [2] of their union $C_i \cup C_j$ on the subspace given by the l_{new} smallest eigenvectors of $C_i \cup C_j$ should be small. Clusters with the smallest energy are being merged until the number of clusters k_c is reduced by a factor α.

ORCLUS detects outliers similarly to PROCLUS. In addition, it discards a certain percentage of seeds srr in each iteration, for which the corresponding clusters contain few points.

ORCLUS inherits the weaknesses of PROCLUS as discussed above.

2.1.2 Hierarchical Approaches

HARP [79] measures the quality of a cluster as the sum of the relevance scores of its relevant attributes. This measure captures the intuition that clusters with as many attributes as possible that are highly relevant are preferable. The relevance score of an attribute a with respect to a set of data points X is computed by comparing the variance of the projections of points in X along attribute a with the variance of the projections of all data points along attribute a.

HARP is an agglomerative, hierarchical clustering technique that starts by placing each data point in a cluster. Two clusters are allowed to merge if the resulting cluster has d_{min} or more relevant attributes, and an attribute is selected as relevant for the merged cluster if its relevance score is greater than R_{min}. d_{min} and R_{min} are two internal thresholds that start at some harsh values so that only points belonging to the same real cluster are likely to be merged. Subsequently, as the clusters increase in size, and the relevant attributes are more reliably determined, the two thresholds are progressively decreased, until they reach some base values or a certain number of clusters has been obtained.

HARP detects outliers by removing small clusters in two stages, as in CURE [38]: first, when the number of clusters reaches a certain fraction of the data set size, and second, near the end of the clustering.

In comparison to partitional approaches to projected clustering, HARP avoids the computation of tentative clusters that may not be reasonable approximations of real clusters. However, HARP is still less effective in the case of low dimensional clusters because of its quality measure. HARP also has the drawback that decisions regarding the clustering of points cannot be undone at a later stage in the algorithm.

---

[2] The energy of a cluster is defined as the sum of squared Euclidean distances between cluster members and the centroid of the cluster.

2.1.3 Density-based Approaches

Density-based approaches to projected clustering can be classified into 1) DBSCAN-like techniques [27]: PreDeCon [18]; 2) hyper-cube-based techniques: DOC/FASTDOC [61], MINECLUS [82], PRIM [30]; and 3) techniques based on the assumption that clusters stand out in low dimensional projections: EPCH [55], FIRES [46].

DBSCAN-like Techniques

PreDeCon [18] computes, given a data point p, a special ε-neighborhood of p, as follows.

First, an ε-neighborhood $N_\varepsilon(p)$ of p using the Euclidean distance in full dimensional space is computed.

Second, p is associated with a *subspace preference vector* w_p so that $w_p[Attr_j] = k$, $k \gg 1$, if the variance of the data points in $N_\varepsilon(p)$ when projected on attribute $Attr_j$ is smaller than a threshold δ; and $w_p[Attr_j] = 1$, in the opposite case. Attributes $Attr_j$ for which $w_p[Attr_j] = k$ are considered relevant attributes for p. The intrinsic dimensionality of p is defined as the number of relevant attributes for p.

Third, PreDeCon defines $dist_p(p,q)$ as the Euclidean distance between data points p and q weighted with the subspace preference vector w_p of p. The intuition behind this distance measure is that data points that are not close to p on the relevant attributes for p are heavily penalized. Since this distance measure is not symmetric, PreDeCon, defines $dist_{pref}(p,q)$ as the maximum between $dist_p(p,q)$ and $dist_q(q,p)$.

Finally, the special ε-neighborhood of p consists of all data points that are within ε distance from p, where the distance used is $dist_{pref}(p,q)$.

Subsequently, a *core* data point is a point with intrinsic dimensionality at most λ, and whose special ε-neighborhood contains at least μ data points. Based on core data points, clusters are defined and found as in the DBSCAN algorithm [27]. The differences from the full dimensional DBSCAN algorithm are that the ε-neighborhood of a data point is modified as described above, and that PreDeCon requires data points that are put in the same cluster to have their intrinsic dimensionality at most λ.

Because of the parameter λ, PreDeCon tends to discover clusters with approximately same dimensionality, and it is sensitive to the numerous parameters required. The computation of relevant attributes is done in full dimensional space, and thus, it is less effective for low dimensional clusters.

Hyper-cube-based Techniques

DOC [61] defines a projected cluster as a pair (X, Y), where X is a subset of data points, and Y is a subset of attributes, such that X contains at least a fraction α of the total number of points, and Y consists of all the attributes on which the projection of X is contained within a segment of length w. DOC uses the function $\mu(|X|, |Y|) = |X| * (1/\beta)^{|Y|}$ to measure the quality of a projected cluster, where β is a user-specified parameter that controls the trade-off between the number of data points and the number of relevant attributes in a projected cluster.

DOC computes one projected cluster at a time. It starts by selecting an arbitrary pivot point p, and subsequently, it randomly selects some data points, different from p, to form a tentative cluster for p. An attribute is considered relevant if the projections of all the tentative cluster's members on this attribute are within distance w from the projection of p on this attribute. The members of the projected cluster are all data points that fall within distance w from p on all attributes deemed as relevant. The process is repeated for $2/\alpha$ pivot points, and for each pivot point, m tentative clusters are tried. Finally, the projected cluster with the highest quality is reported. Then, the technique will be repeated for the next projected cluster. DOC can compute disjoint or overlapping clusters, depending on whether, once a cluster has been found, its points are discarded from the data set or not. Outliers are defined as the points that remain un-clustered. DOC computes values for m and for the size of a tentative cluster so that the method proposed can recover with some high probability projected clusters in the data.

The number of pivot points and tentative clusters that need to be tried can be large. In order to reduce the time complexity of DOC, its authors introduce a variant, called FASTDOC, which uses three heuristics to reduce the search time, but the clustering accuracy is no longer guaranteed. The first heuristic of FASTDOC is to bound the number of tentative clusters tried for a pivot point. The second heuristic is to compute only the relevant attributes for a tentative cluster around a pivot point, and to keep just the largest number of relevant attributes for a tentative cluster that has been observed. Finally, FASTDOC stops when the largest number of relevant attributes for a tentative cluster observed so far is larger than a user-defined d_0.

The performance of DOC is sensitive to the choice of the input parameters, whose values are difficult to determine for real-life data sets. In addition, the assumption that a projected cluster is a hyper-cube of same side length in all attributes may not be appropriate in real applications.

MINECLUS [81] improves upon DOC, by proposing a deterministic method to find the optimal projected cluster centered around a given pivot point p. Each data point is modeled as an itemset that includes the attributes in which the point is within distance w from the pivot point. The problem of finding the projected cluster centered around p with maximum μ value becomes the problem of mining the frequent itemset with the maximum μ value. The paper proposes a technique that modifies a known frequent pattern tree growth method used for mining frequent itemsets. Yet the accuracy of MINECLUS still depends on the three parameters α, β, and w.

To compensate for the effect of these parameters, several heuristic refinement strategies are proposed.

The first heuristic is to add attributes to the set of relevant attributes of a projected cluster. This heuristic covers the case when some relevant attributes have been missed. For this purpose, given a projected cluster (X,Y), MINECLUS computes, for each relevant attribute in Y, a *skew ratio*, which divides the variance of all data points on this attribute by the variance of the cluster members X on this attribute. Attributes which have the skew ratio larger than the minimum skew ratio of all relevant attributes in Y are added to Y.

The second heuristic is to add points to the set of points of a projected cluster. This heuristic covers the case when some cluster points have been missed. For this purpose, given a projected cluster (X,Y), MINECLUS computes, for each point $o \in X$, the Manhattan segmental distance between o and the centroid of X. Points with Manhattan segmental distances to the centroid of X smaller than the maximum Manhattan segmental distances of the points in X are added to X.

The third heuristic removes projected clusters with small μ values, and the final heuristic merge projected clusters that are located closely in similar subspaces until K clusters remain.

PRIM [30] shares some similarities with DOC and its variants because it computes one dense axis-aligned box at a time. Each such box is constructed in a top-down "peeling" phase, which is followed by a bottom-up "pasting" phase. The peeling phase starts with a box B that covers all the data, and at each step, a box b^* is removed from B so that $B \backslash b^*$ contains the largest number of points over all boxes b^* that could have been removed. The peeling phase stops when the current box has less points that a user-defined *mass_min*. A parameter *α_peeling* controls which boxes b^* are candidates for removal. The pasting phase takes the box obtained at the end of the peeling phase, and attempts to extend it with small boxes $\hat{b}$ as long as the number of points in the current box does not decrease. A parameter *α_pasting* controls which boxes $\hat{b}$ are candidates for extension.

Similarly to DOC, the accuracy of PRIM is influenced by the critical parameters *mass_min*, *α_peeling* and *α_pasting*.

Techniques based on low dimensional projections

EPCH [55] computes one or two-dimensional histograms, and "dense" regions are identified in each histogram, as follows. For each histogram, the mean μ and the standard deviation σ of the supports of the cells in the histogram are computed. The cells with support larger than $\mu + c * \sigma$ are declared "dense". Then, dense cells are removed, and the whole process is repeated with the mean and standard deviation of the remaining cells, until there are no more dense cells left or at most $max\_no\_cluster$ times. c and $max\_no\_cluster$ are user-defined parameters. Finally, adjacent dense cells are merged into dense regions.

Projected clusters are computed based on the dense regions. For each data point, a "signature" is derived, which consists of the identifiers of the dense regions the data point belongs to. The similarity between two data points is measured by the matching coefficient of their signatures in which zero entries in both signatures are ignored. Data points are grouped in decreasing order of similarity until at most $max\_no\_cluster$ number of clusters is obtained.

In our experiments, EPCH proved to be very sensitive to the values of its parameters.

FIRES [46] starts with one-dimensional (1D) clusters, called *base* clusters, which can be obtained using any clustering algorithm of choice. To ensure that the base clusters are indeed projections of higher dimensional clusters, base clusters with less points than 25% of the average base cluster size are eliminated. In addition, FIRES uses some heuristics to identify base clusters that contain the projections of more than one higher dimensional cluster. However, these heuristics are less effective in the case of higher dimensional clusters that have more than one 1D projection in common.

FIRES measures the similarity between base clusters as the number of shared points. The base clusters are used to construct a shared k-nearest neighbor graph: vertices correspond to base clusters, and an edge connects two vertices if each vertex is among the k-nearest neighbors of the other vertex. A modified DBSCAN algorithm [26] is applied to this graph. This algorithm takes two user-defined parameters, ϵ and $MinPts$, and produces several sets of base clusters.

Each set of base clusters is used to compute a higher dimensional cluster, as follows: 1) In a "pruning" step, base clusters that produce low quality higher dimensional clusters are removed. The quality of a higher dimensional cluster is a function of its size and dimensionality; 2) DBSCAN is applied on the union of the remaining base clusters.

FIRES computes overlapping clusters. The performance of FIRES is very sensitive to its multiple parameters, namely k for the construction of the shared k-nearest neighbor graph, and ϵ, and $MinPts$ for multiple calls of DBSCAN.

2.2 Subspace Clustering Techniques

Subspace clustering techniques define a subspace cluster as a region in some subspace with density larger than a given density threshold that is surrounded by regions of lower density. The subspace clustering problem is to find all clusters in all subspaces of a data set.

One way of estimating the density of a region in a high dimensional space is to build an axis-aligned grid that partitions the data space into disjoint hyper-rectangular regions, called *units*. A unit is called *dense* if it contains at least some fraction of the points.

In this setting, a subspace cluster is defined as a maximal set of connected dense units in a subset of attributes. Subsequently, the subspace clustering problem is equivalent to the task of automatically identifying subspaces of the original feature space that contain dense units.

Grid-based techniques for subspace clustering suffer from a number of problems. Their performance is typically very sensitive to the resolution of the grid and the density threshold used. Their runtime is exponential in the size of the largest subspace in which clusters exist. They may miss some clusters in cases when heuristic pruning strategies are used. They may miss clusters inadequately oriented or shaped relative to the positioning of the grid.

Another way of estimating the density of a region in a high dimensional space is to generalize the definition of a density-connected cluster underlying the full dimensional clustering algorithm DBSCAN [27] for the problem of subspace clustering. SUBCLU [43] follows this approach, and produces for each subspace the same clusters DBSCAN would have produced, when applied to this subspace. SUBCLU can detect subspace clusters with more general orientation and shape than the grid-based approaches.

A fundamental problem that affects all subspace clustering techniques is the use of global density thresholds for detecting subspace clusters in subspaces of increasing dimensionality. The global density thresholds guarantee some anti-monotonic properties that are used to avoid an exhaustive search through all possible subspaces. However, no meaningful values for these parameters is likely to exist: large values will result in only low dimensional subspace clusters, and small values will results in numerous, spurious low dimensional subspace clusters in addition to higher dimensional subspace clusters.

In the following, we summarize subspace clustering techniques proposed in the literature, and discuss their weaknesses.

CLIQUE [7] overlays an axis-aligned grid over the data space by partitioning each attribute into ξ equi-width units. A unit is *dense* if it contains more than a fraction τ of the points. Both ξ and τ are input parameters.

First, CLIQUE identifies subspaces of the original feature space that contain dense units. Second, for each of the identified subspaces, clusters are computed as disjoint sets of connected dense units. Finally, a description is generated for each cluster by computing its cover with maximal, possibly overlapping, axis-parallel hyper-rectangles.

CLIQUE uses a bottom-up, Apriori-like approach [8] to enumerate dense units in subspaces. 1-dimensional dense units are generated first. *Candidate* k-dimensional units are determined by self-joining $(k-1)$-dimensional dense units that have the first $(k-2)$ attributes in common. The density of a unit can be used to effectively prune the search space, since it is an anti-monotonic property, i.e., a k-dimensional dense unit implies that all its $(k-1)$-dimensional projections are dense. Therefore, candidate k-dimensional units that have $(k-1)$-dimensional projections that are not dense are eliminated. One pass over the data is necessary to determine which of the remaining candidate k-dimensional units are actually dense. The technique terminates when no more candidate units are generated.

The complexity of the dense units generation is exponential in the highest dimensionality of a dense unit. For efficiency reasons, dense units that lie in less "interesting" subspaces are pruned. The interestingness of a subspace is measured by its "coverage", i.e., the fraction of the data points covered by its dense units.

The performance of CLIQUE is very sensitive to the resolution of the grid used, ξ, and the density threshold, τ. The global density threshold, τ, is needed to avoid an exhaustive search through all subspaces. However, it is questionable that a global density threshold is applicable to clusters of increasing dimensionalities, since density decreases as the dimensionality increases. CLIQUE may miss some clusters due to the pruning strategy used, or if the clusters are inadequately oriented or shaped with respect to the positioning of the grid.

nCluster [47] differs from CLIQUE in that the $1D$ units are overlapping windows of length δ. It suffers from the same problems as CLIQUE.

ENCLUS [21] is a grid-based subspace clustering technique that differs from CLIQUE in the criterion used for subspace selection. ENCLUS is based on the observation that the entropy of a subspace is higher when the points are uniformly distributed in the subspace than when the points are closely located in the subspace. A subspace having its entropy below a certain threshold ω is considered "good" for clustering. The entropy of a subspace decreases as the dimensionality of the subspace decreases. Thus, if a k-dimensional subspace has its entropy smaller than ω, then all $(k-1)$-dimensional subspaces obtained by removing one attribute from the k-dimensional subspace have entropy smaller than ω. This anti-monotonic property of entropy is used to generate in a bottom-up, Apriori-like style subspaces that are good for clustering.

ENCLUS suffers essentially from the same problems as CLIQUE. In addition, setting the parameter ω is not very intuitive.

MAFIA [54] is a grid-based subspace clustering technique that addresses some of the drawbacks of CLIQUE. MAFIA partitions each attribute into *adaptive* units that capture the data distribution on that attribute, as follows. Each attribute is divided into a large number of bins. For each bin, the *bin count*, i.e., the number of data points that belong to a bin on an attribute, is computed. Adjacent bins whose bin counts differ by less than a threshold percentage β are merged into units. A single unit on an attribute implies an attribute with uniform distribution. The domain of an attribute with uniform distribution is divided into a fixed number of equi-sized units. A unit on an attribute is dense if it contains α times more points than the expected number of points if the data were uniformly distributed on that attribute.

The number of 1D dense units generated by MAFIA is much smaller than those generated by CLIQUE, which results in a smaller search space than in CLIQUE. In addition, a cluster is represented by a cross-product of dense units, and, thus, MAFIA avoids computing cluster descriptions as in CLIQUE. The dense unit enumeration is similar to CLIQUE, except that no pruning based on "interestingness" is performed. A k-dimensional *candidate* unit is "dense" if it contains α times more points than the expected number of points in any of the $1D$ units that form the k-dimensional unit, where this expected number is computed under the uniform distribution assumption. This definition of density is anti-monotonic, and thus it can be used to efficiently prune the search space. MAFIA only reports "maximal" clusters with respect to this definition of density.

MAFIA still suffers from problems similar to CLIQUE, namely sensitivity to the input parameters and the usage of a global density threshold.

SUBCLU [43] extends the formal definition of a density-connected cluster underlying the DBSCAN algorithm [27] for the problem of subspace clustering. It is shown that the density-connectivity property is anti-monotonic, i.e., if two data points are density-connected in a k-dimensional subspace (with respect to the two input parameters, ϵ and $MinPts$), then the two points are density-connected in any $(k-1)$-dimensional subspace. This property is used to effectively prune the search space in the process of detecting density-connected clusters in all subspaces of a data set.

SUBCLU is able to detect subspace clusters with arbitrary shape and orientation. However, in order to preserve the anti-monotonicity of density-connectivity, it uses global density thresholds (ϵ and $MinPts$), which ignores the fact that data is more sparse as the dimensionality increases.

SCHISM [63] overlays an axis-aligned grid over the data set by partitioning each attribute into ξ intervals of equal width. It defines a "subspace" as an axis-parallel hyper-rectangle formed with cells of the constructed grid. The paper introduces the notion of "interestingness" of a subspace, i.e., a subspace is "interesting" if it contains significantly more points than expected under uniform distribution. This notion of interestingness is materialized in the paper, as follows. If a subspace has dimensionality greater than a certain threshold v, which depends on ξ and the total number of points n, then the subspace is interesting if it contains more points than a constant density threshold, which also depends on ξ and n. Thus, interesting subspaces with dimensionality larger than v can be computed using an Apriori-like search. If a subspace has dimensionality smaller than v, then the subspace is interesting if it contains more points than a variable threshold, which is the minimum between a global density threshold u and another threshold that depends on the dimensionality of the subspace, the total number of points n, and a user-specified significance level τ. Interesting subspaces in the latter category cannot be detected with an Apriori-like algorithm because the variable threshold does not guarantee the anti-monotonic property necessary for an Apriori-like search. The paper proposes a depth-first search heuristic with backtracking that starts from one-dimensional interesting subspaces. However, the method is not guaranteed to recover all interesting subspaces with dimensionality smaller than v. In addition, it is observed that the interesting subspaces are redundant, and thus, the paper proposes to merge similar interesting subspaces, where the similarity is controlled by a user-defined threshold ρ.

The notion of interestingness of a subspace based on statistical principles is valuable. However, for the largest part of the search space, the actual density threshold is a global density threshold, and for the remaining search space, interesting subspace clusters may not be found due to the heuristic search. Also, the interesting subspace clusters found depend on the grid-based discretization of individual attributes.

DUSC [11] uses a density definition based on statistical foundations. DUSC defines a subspace cluster similarly to SUBCLU, except that a point is associated with a density measure, and a point is considered a core point if its density measure is F times larger than the expected value of the density measure under uniform distribution. The definition of a subspace cluster used by DUSC has no anti-monotonic properties, and thus, it cannot be used for pruning the search space. DUSC modifies the definition of a core point so that it has anti-monotonic properties, which, however, introduces a global density threshold.

DiSH [1] is based on the observation that subspace clusters may form hierarchies in which multiple inheritance is possible, i.e., a subspace cluster may be embedded in more than one other subspace cluster. DiSH computes for each point p the highest dimensional subspace in which p fits best. This is achieved by analyzing the ϵ-neighborhood of the point p in each attribute, and keep as "relevant" the attributes where this ϵ-neighborhood contains more than μ points. The relevant attributes are combined bottom-up, in the Apriori style, in order to determine a set of relevant attributes where the ϵ-neighborhood of p contains at least μ points. For efficiency reasons, a best-first search heuristic can be used instead of Apriori.

Subsequently, a distance measure between data points is defined that assigns 1, if both points share a common one-dimensional subspace cluster, 2, if both points share a common two-dimensional subspace cluster, etc. This distance measure is fed into the OPTICS algorithm [10] in order to compute clusters of points. The reachability plot of OPTICS is not suitable for illustrating hierarchies with multiple inclusions; thus, DiSH includes a method and a visualization tool for this task.

Similarly to the other techniques in this category, DiSH uses a bottom-up strategy and relies on a global density threshold to determine the subspace associated with a data point.

2.3 Categorical Subspace and Projected Clustering

Existing techniques for subspace and projected clustering discussed in the previous sections are designed for *numerical* data sets, i.e., data sets where the domain of every attribute is inherently ordered. However, many real data sets are *categorical*, i.e., the attribute domains are discrete and not ordered.

Categorical subspace and projected clustering is the task of subspace, respectively, projected clustering applied to categorical data sets.

Subspace and projected clustering techniques designed for numerical data are not readily applicable to categorical data sets. Some of these techniques (e.g., PROCLUS) use distance functions that exploit the geometric properties of the data space, which cannot be effectively captured by categorical distance functions, such as the simple matching coefficient. Other techniques (e.g., HARP) require numerical computations that are not well-defined for categorical attributes (e.g., mean, variance). Finally, some techniques (e.g., CLIQUE) are based on the discretization of individual data attributes into bins, and the notion of "neighboring" bins is used to manage the search through all possible subspaces. Categorical attributes lack order, and thus this notion is not directly applicable.

A significantly smaller body of work has been dedicated to the subspace and projected clustering problems on categorical data than to the same problems on numerical data. To the best of our knowledge, the only techniques proposed for subspace and projected clustering of categorical data are SUBCAD [32], CLICKS [84], and to some extent STIRR [35] and CACTUS [33]. STIRR does not specify how to aggregate clusters based on their projections on individual attributes. CACTUS can mine only a limited class of subspace clusters. Common weaknesses of SUBCAD and CLICKS are that their accuracy depends heavily on parameters that are difficult to set appropriately and/or on initialization strategies in full dimensional space.

STIRR [35] models a categorical data set using a graph structure in which vertices correspond to individual attribute values, and an edge exists between two vertices if they co-occur in some data point. A *basin* represents a set of weights assigned to the vertices in this graph. STIRR iterates multiple basins of this graph by propagating weights via co-occurrence until the basins eventually converge to a fixed point. The authors argue that, once the fixed point is reached, the weights in the basins can be used to separate cluster projections on attributes. However, the separation of attribute values based on their weights is not intuitive, and it was shown to produce unsatisfactory results in the case of clusters with overlapping projections [33]. More importantly, even if the cluster projections on individual attributes have been identified correctly, the technique does not specify how to aggregate the projections into clusters.

Zhang et al [86] notice that there are cases in which STIRR does not converge, and introduce a similar technique that is guaranteed to converge. However, this technique suffers from the same problems as STIRR.

SUBCAD [32] is a projected clustering technique for categorical data that aims at partitioning a data set into a user-specified K number of projected clusters such that a certain objective function is minimized. SUBCAD is initialized by selecting K scattered seeds and assigning each data point to the closest seed, where distance is measured using the simple matching coefficient in full dimensional space. Cluster membership and relevant attribute selection are guided by minimization of the objective function. The performance of SUBCAD is very sensitive to its initialization. In addition, SUBCAD has no mechanism for detecting outliers.

CACTUS [33] models a categorical data set as a graph, in which vertices correspond to individual attribute values, and an edge exists between a pair of attribute values from different attributes if their support is α times larger than their expected support. Expected supports, both in CACTUS and later, in CLICKS, are computed based on the uniform distribution assumption. Attribute values on the same attribute are connected if they are linked to a common attribute value, on a different attribute, in the graph representation.

First, CACTUS computes, for each attribute, all cluster projections on it, as it will be explained shortly. Second, cluster projections on individual attributes are used to generate cluster candidates of higher dimensionality. Due to the level-wise cluster candidates generation technique, CACTUS discovers only a limited class of subspace clusters, i.e., clusters in the subspaces $(Attr_1)$, $(Attr_1, Attr_2)$, $(Attr_1, Attr_2, Attr_3)$, $\ldots$,$(Attr_1, \ldots Attr_d)$. Cluster candidates with support α times larger than their expected support are reported as clusters.

CACTUS computes cluster projections on individual attributes, in two steps: 1) each attribute $Attr_i$ is clustered with respect to every other attribute $Attr_j$, $i \neq j$, to find all cluster projections on $Attr_i$ of clusters over $(Attr_i, Attr_j)$, and 2) cluster projections on $Attr_i$ of clusters over $(Attr_1, \ldots, Attr_d)$ are computed by intersecting the projections found in step 1). Step 1) is performed by computing, on each attribute, "distinguishing" sets of size at most k. A distinguishing set is a set of attribute values that uniquely occur within only one cluster. Distinguishing sets of size at most k are computed as cliques of size at most k.

The distinguishing sets are based on the assumption that clusters are uniquely identified by a core of attribute values that do not occur in other clusters. This assumption is not necessarily true in all data sets. Furthermore, k should be chosen as small as possible in order to detect all clusters. However, a small k produces a large number of cluster projections on individual attributes, which in turn causes a very large number of candidate clusters to be generated. The accuracy of CACTUS highly depends on its parameters, k and α.

CLICKS [84] is based on the same graph representation of a data set as CACTUS. In order to detect clusters whose projections on attributes consist of more than one attribute value, all values of an attribute are considered to be implicitly connected. Subspace clusters correspond to dense, maximal *cliques* in the associated graph. A clique is considered *dense* if its support is α times larger than its expected support.

In a preprocessing step, CLICK computes the graph representation for a categorical data set. Subsequently, all maximal cliques are detected on this graph using a recursive algorithm. Two additional post-processing steps are performed: 1) dense maximal cliques and dense maximal sub-cliques of a non-dense maximal clique are computed, and 2) the cliques computed in step 1) are merged if they share more than σ points. Merges are performed in a certain order, determined by the coverage of cliques being merged.

The accuracy of CLICKS is highly dependent on the values of its parameters, α and σ.

2.4 Related Problem Formulations

In this thesis, we consider the problem formulation in which a subspace is defined as a subset of the original attributes of a data set. For this reason, the techniques in this category are sometimes called *axis-parallel* subspace clustering techniques.

A related problem formulation is one in which a subspace is defined as an arbitrary set of orthogonal vectors [6].

A (linear) correlation (subspace) cluster is defined as a subset of points that form a λ-dimensional hyper-plane, where λ is smaller than the data dimensionality d [19]. The key property of a correlation cluster is that Principal Component Analysis (PCA) on the cluster points will reveal several eigenvalues that are "smaller" than the rest. Equivalently, the points of a correlation cluster are closely located in the subspace given by the eigenvectors corresponding to the "small" eigenvalues, and this subspace is orthogonal to the λ-dimensional hyper-plane where cluster points reside. For this reason, a correlation cluster is sometimes called a generalized projected cluster. The difference between a projected cluster and a linear correlation cluster is that the "small" eigenvectors of a projected cluster are axis-parallel, whereas the "small" eigenvectors of a correlation cluster can have arbitrary orientation.

Projected clustering techniques cannot in general recover correlation clusters because the points of a correlation cluster are not necessarily closely located in their relevant subspace. For the same reason, correlation clustering techniques cannot in general recover projected clusters, except ORCLUS, due to its similarity to PROCLUS. Existing subspace clustering techniques are based on detecting dense low dimensional projections of clusters and aggregating them bottom-up; thus, these techniques cannot recover correlation clusters unless the correlation clusters have dense low dimensional projections, which may not be a realistic assumption.

Representative correlation clustering techniques are ORCLUS [6], 4C [19] and COPAC [2]. These techniques have several drawbacks. First, they are based on the "locality assumption", i.e., that a full dimensional neighborhood of a cluster point reflects the hyper-plane on which this point resides. The locality assumption means that, in order for these techniques to work, a full dimensional neighborhood of a cluster point should be dominated by cluster points. However, in this case, the full dimensional clustering algorithms are likely to perform well. Second, these algorithms often require parameters that are hard to set, such as the number of clusters, or the cluster dimensionality λ. Finally, some of these techniques are restrictive in the sense that they require all clusters to have the same dimensionality.

A generalization of (linear) correlation clustering is *non-linear* correlation clustering. A *non-linear* correlation (subspace) cluster is a subset of points non-linearly correlated in a subspace. PCA is not effective for detecting non-linear correlations, and thus other techniques, such as the fractal dimension, are needed to compute the intrinsic dimensionality of a subset of points. Representative techniques are CURLER [66] and DIC [36]. Detecting non-linear correlations is more difficult than detecting linear correlations because of various non-linear relationships that could exist in the data. Existing techniques suffer from the same problems as the correlation clustering techniques.

COSA [31] can be seen as a problem formulation related to the projected clustering problem: whereas, in projected clustering, each cluster exists in a certain subspace, in COSA, each cluster exists in all attributes, but each attribute has a certain weight with respect to each cluster, which reflects the "relevance" of an attribute with respect to a cluster. The task of the clustering algorithm is to compute a partition of the data points into several clusters, as well as the attribute weights associated with a cluster. This task can be cast as the problem of optimizing a certain objective function, which in this case is a complicated non-convex function with numerous local minima. Since no direct method for optimizing this objective function has been found, COSA proposes an approximate method for determining simultaneously the clusters and the attribute weights associated with each cluster.

Another related problem formulation that has emerged within the bioinformatics community is biclustering [50]. A bicluster is a pair (X, Y), where X is a subset of data points, and Y is a subset of data attributes, so that a certain homogeneity criterion is satisfied. The homogeneity criterion is usually symmetric in terms of rows and columns, and from here the term *bi*clustering. Biclustering algorithms usually find maximal (X, Y) that satisfy the homogeneity criterion, and often require that $|X| \geq min_o$, and $|Y| \geq min_a$, where min_o, min_a are user-defined parameters.

Several homogeneity criteria have been proposed in the literature:

- Biclusters with constant values. They correspond to axis-parallel projected/subspace clusters.
- Biclusters with constant values on columns. They correspond to axis-parallel projected/subspace clusters.
- Biclusters with constant values on rows. The points of such a bicluster reside on the bisecting line in its relevant subspace.
- Biclusters with coherent values. The points of such a bicluster reside on a line with positive slope in its relevant subspace. This bicluster model subsumes the aforementioned three models. Representative algorithms are delta-clusters [22], FLOC [77], CoClus [24], pCluster [69], [58], [68], [87].
- Biclusters with coherent evolution. In such a bicluster, every data point induces the same linear ordering of its attributes based on the values that the data point takes in these attributes. This problem is equivalent to sequential pattern mining. Representative algorithms are [14], [48], [23], [34].
- *Reg-clusters* are biclusters that combine the characteristics of biclusters with coherent evolution with the characteristic of biclusters with coherent values [76].

For a comprehensive survey on biclustering, the reader is referred to [50]. We note that correlation clustering is a more general problem formulation than biclustering, because biclustering is limited to a special form of correlation where the attributes are positively correlated.

Co-clustering models a data matrix as a weighted bipartite graph, where vertices correspond to data points and data attributes, and edges between them are weighted by the corresponding entries in the data matrix. Co-clustering applies spectral clustering techniques to this bipartite graph, and derives a partition of data points and data attributes into K groups so that the weight of the within-groups edges is maximized and the weight of the between-groups edges is minimized. Co-clustering is a restricted version of subspace clustering because an attribute can belong to at most one group, i.e., co-clustering cannot produce clusters of points that exist in overlapping subspaces.

Related to our work is also the work on Scan Statistics [4], in which the goal is to detect spatial regions for which their z-score (i.e., the number of standard deviations by which the observed count of some variable of interest is higher than the expected count of the variable of interest in a spatial region) is significantly high, given the distribution of the maximum z-score of all the regions under the null hypothesis of no clusters. The methods in Scan Statistics are applicable to full dimensional data, whereas our work will be concerned with statistically significant regions in any/all subspaces of a data set.

Chapter 3

P3C: Projected Clustering via Cluster Cores

We have identified in our literature survey several drawbacks of existing subspace and projected clustering techniques. Concerning subspace clustering techniques, we have observed that: 1) they are based on global density thresholds for which no meaningful value is likely to exist; 2) they report a large number of overlapping clusters; and 3) the grid-based techniques are sensitive to the grid resolution. Concerning projected clustering techniques, we have observed that: 1) they rely greatly on parameters whose appropriate values are difficult to anticipate by the users; 2) they are unable to identify clusters with few relevant attributes; and 3) most of them restrict the membership of a data point to at most one cluster.

In this work, we assume the following definition of a projected cluster, which is also used by recent projected clustering work (e.g., SSPC [80], HARP [79], EPCH [55] and FIRES [46]).

Definition 3.1 A projected cluster is a pair (X, Y), where X is a subset of data points and Y is a subset of attributes so that 1) the points in X project along each attribute $a \in Y$ on a "small" range of values, compared to the range of values on which the whole data set projects on a, and 2) the points in X are uniformly distributed along each attribute a' *not* in Y.

The notion of "small" range is defined implicitly in our work, as described below in Section 3.3.

We design a new technique for projected clustering with the following goals in mind:

- The technique should effectively discover clusters in the data while requiring as few parameters as possible. Moreover, setting these parameters should require minimal prior knowledge about the data, and the technique should be robust with respect to these parameters.
- The technique should be able to discover low dimensional clusters embedded in high dimensional spaces.

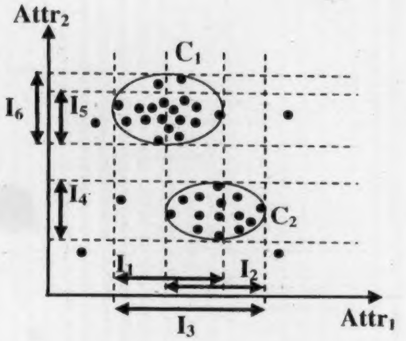

Figure 3.1: Two 2D projected clusters

- The technique should be able to compute overlapping clusters, i.e., clusters that share data points.
- The technique should be robust to noise.

In the following, we present the technique P3C (Projected Clustering via Cluster Cores) that satisfies the goals outlined above. To present our technique, we introduce some definitions and notation in Section 3.1.

3.1 Preliminary Definitions

Let $D = \{(x_{i1}, \ldots, x_{id}) | 1 \leq i \leq n\}$ be a data set of n d-dimensional data points. Let $A = \{Attr_1, \ldots, Attr_d\}$ be the set of the d attributes of the points in D so that $x_{ij} \in dom(Attr_j)$, where $dom(Attr_j)$ denotes the domain of the attribute $Attr_j$, $1 \leq j \leq d$. Without restricting the generality, we assume that all attributes have normalized domains, i.e., $dom(Attr_j) = [0, 1]$, and we also refer to projections of a point $x_i \in D$ using dot-notation, i.e., if $x_i = (x_{i1}, \ldots, x_{id})$ then $x_i.Attr_j = x_{ij}$.

An ***interval*** $I = [v_l, v_u]$ on an attribute $a \in A$ is defined as all real values $x \in dom(a)$ so that $v_l \leq x \leq v_u$. The width of interval I is defined as $width(I) := v_u - v_l$. The associated attribute of an interval I is denoted by $attr(I)$.

Figure 3.1 illustrates a data set with two projected clusters, C_1, and C_2, both having $Attr_1$ and $Attr_2$ as the only relevant attributes. Equivalently, the points in C_1 and C_2 are uniformly distributed on all other attributes that the data set may have. I_1, I_2, and I_3 are intervals on attribute $Attr_1$; I_4, I_5 and I_6 are intervals on attribute $Attr_2$; $attr(I_1) = attr(I_2) = attr(I_3) = Attr_1$, and $attr(I_4) = attr(I_5) = attr(I_6) = Attr_2$.

A ***subspace*** S is a non-empty subset of attributes, $S \subseteq A$. The dimensionality of S, $dim(S)$, is the cardinality of S.

A ***hyper-rectangle*** H is an axis-aligned box of intervals on different attributes in A, $H = I_1 \times \ldots \times I_p$, $1 \leq p \leq d$, and $attr(I_i) \neq attr(I_j)$ for $i \neq j$. $S = \{attr(I_1), \ldots, attr(I_p)\}$ is the subspace of H, denoted by $subspace(H)$. I_j is also called the projection of H on attribute $attr(I_j)$, $1 \leq j \leq p$. The intervals of H are denoted by $intervals(H) := \{I_1, \ldots, I_p\}$.

For example, in Figure 3.1, $H = I_3$ x I_4 is a hyper-rectangle in subspace $S = \{Attr_1, Attr_2\}$, where I_3 is the projection of H on attribute $Attr_1$, and I_4 is the projection of H on attribute $Attr_2$. I_3 x I_1 is not a hyper-rectangle, because I_3 and I_1 are intervals on the same attribute $Attr_1$.

Let $H = I_1 \times \ldots \times I_p$ be a hyper-rectangle, $1 \leq p \leq d$. The ***volume*** of H, denoted by **vol(H)**, is defined as the hyper-volume occupied by H in $subspace(H)$, which is computed as $vol(H) = \prod_{i=1}^{p} width(I_i)$. The ***support set*** of H, denoted by **SuppSet(H)**, represents the set of database points whose coordinate values fall within the intervals of H for the corresponding attributes in $subspace(H)$, i.e., $SuppSet(H) := \{x \in D | x.attr(I_i) \in I_i, \forall i : 1 \leq i \leq p\}$. The ***actual support*** of H, denoted by **AS(H)**, represents the cardinality of its support set, i.e., $AS(H) := |SuppSet(H)|$.

A ***characteristic hyper-rectangle*** $\tilde{H}$ of a projected cluster (X, Y), $X \subseteq D$, $Y \subseteq A$, $|Y| = p$, is a hyper-rectangle $\tilde{H} = I_1 \times \ldots \times I_p$, where I_j is the smallest interval on attribute $attr(I_j)$ that contains the projections on $attr(I_j)$ of all the points in X, $1 \leq j \leq p$.

In Figure 3.1, the characteristic hyper-rectangle of projected cluster C_1 is the hyper-rectangle $\tilde{H_1} = I_1 \times I_6$, and the characteristic hyper-rectangle of projected cluster C_2 is the hyper-rectangle $\tilde{H_2} = I_2 \times I_4$.

Since an attribute may be relevant to more than one projected cluster, characteristic hyper-rectangles may contain overlapping intervals. In Figure 3.1, intervals I_1 and I_2 overlap on attribute $Attr_1$. We assume that characteristic hyper-rectangles can contain overlapping intervals as long as they are not completely *nested* within each other. Characteristic hyper-rectangles $\tilde{H}$ and $\tilde{L}$ are *nested* if for every interval I_i in $\tilde{H}$, there is an interval I_j in $\tilde{L}$ so that $I_i \subseteq I_j$.

3.2 Overview of P3C

P3C is based on the idea that if the characteristic hyper-rectangles of projected clusters were known, then clusters can be immediately computed as the support sets of the characteristic hyper-rectangles. Since the characteristic hyper-rectangles are not known, P3C computes in two steps a set of hyper-rectangles that match or approximate well the characteristic hyper-rectangles of projected clusters in the data. First, on every attribute, intervals that match or approximate well projections of characteristic hyper-rectangles on that attribute are computed (Section 3.3). Second, the challenge is to determine which intervals actually represent the same characteristic hyper-rectangle. P3C addresses this challenge by aggregating the computed intervals into *cluster cores*. Roughly

```
Input: Data set $D = \{(x_{i1}, \ldots, x_{id}) | 1 \leq i \leq n\}$, parameter $\alpha_{Binom}$.
Output: Several disjoint or overlapping clusters, their relevant attributes, and outliers.
Method:
    1. For each attribute, compute one-dimensional intervals that approximate projections of characteristic hyper-rectangles on that attribute (Section 3.3).
    2. Aggregate intervals computed in step 1 into cluster cores (Section 3.4).
    3. Refine cluster cores into clusters (either disjoint or overlapping), compute outliers, and refine relevant attributes of clusters (Section 3.5).
```

Figure 3.2: Pseudo-code of P3C

speaking, a cluster core is a hyper-rectangle H that approximates well a characteristic hyper-rectangle $\tilde{H}$ of a projected cluster C in the sense that a large fraction of the points in $SuppSet(H)$ belongs to C (Section 3.4).

For the example in Figure 3.1, P3C first computes the interval I_3 on attribute $Attr_1$ that approximates the projections of the characteristic hyper-rectangles $\tilde{H}_1 = I_1 \times I_6$ and $\tilde{H}_2 = I_2 \times I_4$ on attribute $Attr_1$, and intervals I_5 and I_4 that approximate/match the projections of the same characteristic hyper-rectangles on attribute $Attr_2$. Second, P3C aggregates these intervals into two cluster cores, i.e., $H_1 = I_3 \times I_4$ and $H_2 = I_3 \times I_5$, which can be regarded as approximations of the two projected clusters in the data.

Cluster cores may include in their support sets additional points that do not belong to the projected clusters that they approximate. This happens when the intervals are wider than the projections of characteristic hyper-rectangles that they approximate. In Figure 3.1, interval I_3 is wider than interval I_2, and thus, the support set of cluster core $H_2 = I_3$ x I_4 includes points that do not belong to cluster C_2. On the other hand, cluster cores may not include completely in their support sets the projected clusters that they approximate. This is the case when the intervals are tighter than the projections of characteristic hyper-rectangles that they approximate. In Figure 3.1, interval I_5 is tighter than interval I_6, and thus the support set of cluster core $H_1 = I_3$ x I_5 does not include all points of cluster C_1. Thus, in order to compute the projected clusters, the supports sets of cluster cores are refined, outliers are detected, and relevant attributes for each cluster are refined (Section 3.5).

The pseudo-code of P3C is given in Figure 3.2.

3.3 Approximating Projections of Characteristic Hyper-rectangles

By Definition 3.1 of a projected cluster, an attribute that is irrelevant for all projected clusters exhibits uniform distribution. In contrast, an attribute that is relevant for at least one projected cluster will exhibit in general a non-uniform distribution, because it contains one or more intervals with unusual high support corresponding to projections of clusters on that attribute.

Consequently, we need to identify attributes with uniform distribution, and, for the non-uniform attributes, to identify intervals with unusual high support. For this task, the *Chi-square* goodness-of-fit test [65] is employed.

The Chi-square test uses the methodology of statistical hypothesis testing, which is summarized in appendix A. The null hypothesis is that the n data points are uniformly distributed on an attribute. In order to compute the test statistic of the Chi-square test, each attribute is divided into the same number of equi-width bins. Sturge's rule [65] suggests that the number of bins should be equal to $\lfloor 1 + log_2(n) \rfloor$. For every bin in each attribute, its support is computed. The Chi-square test statistic sums, over all bins in an attribute, the squared difference between the bin support and the average bin support, normalized by the average bin support. The distribution of the Chi-square test statistic under the null hypothesis is a Chi-square distribution with $(no\_bins - 2)$ degrees of freedom [65], where no_bins is the number of bins on an attribute, and $no\_bins = \lfloor 1 + log_2(n) \rfloor$.

Let α_{Chi}^{Unif} be a significance level, and let $\theta_{\alpha_{Chi}^{Unif}}$ be the right critical value of the Chi-square distribution with $(no\_bins - 2)$ degrees of freedom at significance level α_{Chi}^{Unif}. $\theta_{\alpha_{Chi}^{Unif}}$ can be found in pre-computed tables or, if needed, can be approximated numerically. Attributes for which the Chi-square test statistic is less than $\theta_{\alpha_{Chi}^{Unif}}$ are reported as uniform, whereas attributes for which the Chi-square test statistic is greater than $\theta_{\alpha_{Chi}^{Unif}}$ are reported as non-uniform. Equivalently, the probability of declaring an attribute non-uniform when in fact the attribute is uniform is very small, i.e., less than α_{Chi}^{Unif}.

On the attributes deemed non-uniform, the bin with the largest support is *marked*. The remaining un-marked bins are tested again using the Chi-square test for uniform distribution. If the Chi-square test indicates that the un-marked bins "look" uniform, then we stop. Otherwise, the bin with the second-largest support is marked. Then, we repeat testing the remaining un-marked bins for the uniform distribution and marking bins in decreasing order of support, until either the current set of un-marked bins satisfies the Chi-square test for uniform distribution or there are two un-marked bins left. The Chi-square test can be applied as long as the current number of un-marked bins on an attribute is at least 3, because the distribution of the test statistic is a Chi-square distribution with $(no\_bins - 2)$ degrees of freedom. If the Chi-square test has been applied repeatedly on an attribute until two un-marked bins are left on that attribute, then the bin with larger support is marked.

At this point, intervals on each attribute are computed by merging adjacent marked bins. These intervals approximate the projections of characteristic hyper-rectangles on their attributes. Let $\mathcal{I}$ be the set of all intervals computed in this manner on all attributes.

Because *adjacent* marked bins are merged into intervals, the resulting intervals capture implicitly the fact that the points of a projected cluster are located relatively closely when projected on relevant attributes of the projected cluster.

The computed intervals may be wider or tighter than the projections of characteristic hyper-rectangles that they approximate. Characteristic hyper-rectangles that contain overlapping intervals may lead to the former case (e.g., in Figure 3.1, intervals I_1 and I_2 are approximated by interval I_3). An example of the latter case is an interval that approximates the projection of a characteristic hyper-rectangle on an attribute where the cluster is Gaussian distributed. In this case, the interval may only capture the most dense region of the projection (e.g., in Figure 3.1, interval I_5 on attribute $Attr_2$).

3.4 Cluster Cores Computation

In Figure 3.1, the computed intervals form only two possible hyper-rectangles, $H_1 = I_3 \times I_5$ and $H_2 = I_3 \times I_4$, which actually represent the two projected clusters C_1 and C_2. However, in practical applications, the number of possible hyper-rectangles that can be constructed from the set of computed intervals $\mathcal{I}$ is large. The challenge is to determine which hyper-rectangles do in fact represent projected clusters. This section describes how P3C addresses this challenge.

Let $\tilde{T} = I_1 \times \ldots \times I_p \times I_{p+1} \times \ldots \times I_t$, $1 \leq p < t \leq d$, be a characteristic hyper-rectangle of some projected cluster. Let $H = J_1 \times \ldots \times J_p$ be a hyper-rectangle so that each J_i approximates I_i, $1 \leq i \leq p$. Let I' be an interval so that $attr(I')$ does not belong to $subspace(H)$. Let $H' = J_1 \times \ldots \times J_p \times I'$.

We would like to determine if I' approximates one of the intervals $I_{p+1}, \ldots, I_t$. Let us consider the case when interval I' does ***not*** approximate one of the intervals $I_{p+1}, \ldots, I_t$. In this case, we want to compute how many points are expected to be in H', i.e., we want to compute the expected value of $AS(H')$. Clearly, the actual support $AS(H')$ of H' is equal to the number of points in $SuppSet(H)$ that also belong to I'.

Because J_i approximates I_i, $1 \leq i \leq p$, the points in $SuppSet(H)$ are mainly points of a projected cluster with characteristic hyper-rectangle $\tilde{T}$. When interval I' does ***not*** approximate one of the intervals $I_{p+1}, \ldots, I_t$, we can assume, by the definition of a projected cluster, that the points in $SuppSet(H)$ are uniformly distributed on the attribute $attr(I')$ of interval I'.

The null hypothesis is that the points in $SuppSet(H)$ are uniformly distributed on the attribute $attr(I')$ of interval I'. The test statistic is $AS(H')$ or, equivalently, the number of points in $SuppSet(H)$ that also belong to I'. We need to determine the distribution of the test statistic $AS(H')$ under the null hypothesis.

The Binomial distribution with parameters x and $prob$ is the discrete probability distribution of the number of successes in a sequence of x independent "yes/no" experiments, each of which yields success with probability $prob$. In our case, an experiment consists of assigning to a data point from $SuppSet(H)$ an attribute value on $attr(I')$ that is uniformly distributed on $attr(I')$. The outcome of each experiment is either "yes", when the data point belongs to I', or "no", in the opposite case. There are $AS(H)$ "yes/no" experiments, because there are $AS(H)$ data points in $SuppSet(H)$, and the experiments are mutually independent. The probability of success in each experiment, i.e., the probability that the data point belongs to I', is $\frac{vol(I')}{vol(attr(I'))} = \frac{width(I')}{width(attr(I'))}$. Since we assume normalized data to $[0,1]$, the probability of success in each experiment is $width(I')$ [1]. Therefore, the distribution of the test statistic $AS(H')$ under the null hypothesis is the Binomial distribution with parameters $AS(H)$ and $width(I')$.

The mean of the Binomial distribution with parameters $AS(H)$ and $width(I')$ is $AS(H) * width(I')$, and represents the expected value of $AS(H')$ or, equivalently, the expected number of points in $SuppSet(H)$ that also belong to I', under the null hypothesis that the points in $SuppSet(H)$ are uniformly distributed on attribute $attr(I')$ of interval I'.

Let α_{Binom} be a significance level, and let $\theta_{\alpha_{Binom}}$ be the right critical value of the Binomial distribution with parameters $AS(H)$ and $width(I')$ at significance level α_{Binom}. $\theta_{\alpha_{Binom}}$ can be computed exactly numerically, because the Binomial distribution is a discrete distribution.

Under the null hypothesis that the points in $SuppSet(H)$ are uniformly distributed on the attribute $attr(I')$ of interval I', the probability that $AS(H') \geq \theta_{\alpha_{Binom}}$ is very small, i.e., less than α_{Binom}. Thus, we consider that if $AS(H') \geq \theta_{\alpha_{Binom}}$, then this is evidence that the null hypothesis can be rejected. Therefore, we consider that if $AS(H') \geq \theta_{\alpha_{Binom}}$, then this is evidence that in fact I' approximates one of the intervals $I_{p+1}, \ldots, I_t$.

To summarize, given a hyper-rectangle $H = J_1 \times \ldots \times J_p$ formed with intervals computed in Section 3.3 (i.e., $intervals(H) \subseteq \mathcal{I}$), and an interval $I' \in \mathcal{I}$ so that $attr(I')$ does not belong to $subspace(H)$, we add I' to H and obtain the hyper-rectangle $H' = J_1 \times \ldots \times J_p \times I'$ only if $AS(H') \geq \theta_{\alpha_{Binom}}$, where $\theta_{\alpha_{Binom}}$ is the right critical value of the Binomial distribution with parameters $AS(H)$ and $width(I')$ at a significance level of α_{Binom}. In this case, we say that there is *evidence* that I' approximates the same projected cluster as H.

---

[1] We note that assuming normalized data to $[0,1]$ is not a restriction of the method; for non-normalized data, the probability of success in each experiment shall be computed as $\frac{width(I')}{width(attr(I'))}$.

A hyper-rectangle H approximates the characteristic hyper-rectangle of a projected cluster if H consists of 1) *only* and 2) *all* intervals that approximate the projections of the cluster on its relevant attributes. The first condition is equivalent to requesting that for any hyper-rectangle $Q = I_{Q_1} \times \ldots \times I_{Q_q}$, where $q \geq 1$ and $intervals(Q) \subset intervals(H)$, and for any interval $I' \in intervals(H) \setminus intervals(Q)$, there is evidence that I' approximates the same projected cluster as Q. The second condition is equivalent to requesting that H is *maximal*, i.e., for any interval $I' \in \mathcal{I}$ so that $attr(I')$ does not belong to $subspace(H)$, there is *no* evidence that I' approximates the same projected cluster as H. Formally, a cluster core can be defined as following.

Definition 3.2 Let α_{Binom} be a significance level. A *cluster core* is a hyper-rectangle H with the following properties:

1. For any hyper-rectangle $Q = I_{Q_1} \times \ldots \times I_{Q_q}$, where $q \geq 1$ and $intervals(Q) \subset intervals(H)$, and for any interval $I' \in intervals(H) \setminus intervals(Q)$, it holds that:
   $AS(Q \times I') \geq \theta_{\alpha_{Binom}}$
   where $\theta_{\alpha_{Binom}}$ is the right critical value of the Binomial distribution with parameters $AS(Q)$ and $width(I')$ at significance level α_{Binom}.

2. For any interval $I' \in \mathcal{I}$ so that $attr(I')$ does not belong to $subspace(H)$, it holds that:
   $AS(H \times I') < \theta_{\alpha_{Binom}}$
   where $\theta_{\alpha_{Binom}}$ is the right critical value of the Binomial distribution with parameters $AS(H)$ and $width(I')$ at significance level α_{Binom}.

Condition 1 in Definition 3.2 is anti-monotonic in the sense that, given a hyper-rectangle H that satisfies Condition 1, any hyper-rectangle formed with a subset of the intervals in H will also satisfy Condition 1. This fact motivates an Apriori-like generation of hyper-rectangles that satisfy Condition 1: a hyper-rectangle that consists of $(q + 1)$ intervals will not be generated unless all hyper-rectangles that can be obtained with q intervals out of these $(q + 1)$ intervals have been already generated. Hyper-rectangles that satisfy Condition 1 are generated and the ones that are "maximal" in the sense of Condition 2 are reported as cluster cores.

3.5 Cluster Cores Refinement, Outlier Detection, Relevant Attributes Refinement

Let K be the number of cluster cores constructed according to Section 3.4. A cluster core is a hyper-rectangle that approximates a projected cluster in the data in the sense that it contains a large fraction of the points of the projected cluster. However, the support set of a cluster core may not necessarily contain all and only the points of the projected cluster approximated by the cluster core. In addition, the support sets of the K cluster cores are not necessarily disjoint, because data points may belong to more than one cluster core. Also, some data points may not belong to any cluster core.

We want to refine the computed cluster cores into clusters, to determine outliers, and to refine the relevant attributes of each cluster. The first two tasks are performed in a subspace of (reduced) dimensionality $d' \leq d$, containing all attributes that were deemed non-uniform according to the analysis presented in Section 3.3.

Let a projected cluster be represented by its mean μ and its covariance matrix Σ. By Definition 3.1 of a projected cluster, the points of the cluster project within a "small" range on the relevant attributes, and are uniformly distributed on the irrelevant attributes. Therefore, given the eigenvectors and the eigenvalues of the covariance matrix Σ, some of these eigenvalues will be "small" in comparison with the remaining eigenvalues, and the eigenvectors corresponding to the "small" eigenvalues correspond to the relevant attributes of the cluster. Moreover, cluster members project closely to the cluster mean on the eigenvectors corresponding to the "small" eigenvalues.

The Mahalanobis distance between a data point x and a cluster mean μ is defined as $MahalanobisDist(x, \mu) := ((x-\mu)^T * \Sigma^{-1} * (x-\mu))^{1/2}$. Equivalently, $MahalanobisDist(x, \mu) = (\sum_{j=1}^{d'} \frac{z_j^2}{\lambda_j})^{1/2}$, where λ_j is the j^{th} eigenvalue of Σ, and z_j is the projection of $(x - \mu)$ on the j^{th} eigenvector e_j. The distances between x and μ on the "small" eigenvectors dominate the computation of the Mahalanobis distance, because these distances are down-weighted by a small factor λ_j, whereas the distances between x and μ on the remaining eigenvectors count less in the computation of the Mahalanobis distance, since they are down-weighted by a large factor λ_j.

Thus, cluster members have shorter Mahalanobis distances to cluster means than non-cluster members. Based on these considerations, we assign data points that do not belong to any cluster core to the cluster core with the "closest" mean in terms of Mahalanobis distances. At this point, the K cluster cores correspond to a fuzzy partitioning of the data set into K clusters, which is used to initialize the Expectation Maximization (EM) algorithm [25].

The EM algorithm iteratively repeats two steps: an E-step, in which the current clusters are refined, by computing the membership probabilities of data points to clusters, based on Mahalanobis distances between data points and cluster means, and a M-step, in which means and covariance matrices of the current clusters are computed. EM stops when the means of the computed clusters remain unchanged between two consecutive iterations. When starting with cluster cores, it typically takes only 5 to 10 iterations until convergence, since the cluster cores typically approximate well projected clusters in the data.

The output of EM is a matrix of probabilities that gives for each data point its probability of belonging to each cluster. If disjoint clusters are desired, each data point is assigned to the most probable cluster. If overlapping clusters are needed, a data point is assigned to a cluster if the probability of belonging to it is larger than $1/K$.

Clustering and outlier detection are closely related. We use a standard technique for multi-variate outlier detection [65]. The Mahalanobis distances between data points and the means of the clusters to which they belong are compared to the right critical value $\theta_{\alpha_{Chi}^{Outl}}$ of the Chi-square distribution with d' degrees of freedom at a significance level of α_{Chi}^{Outl}. The critical value $\theta_{\alpha_{Chi}^{Outl}}$ can be found in pre-computed tables or approximated numerically, if needed. Data points with Mahalanobis distances to cluster means larger than $\theta_{\alpha_{Chi}^{Outl}}$ are declared outliers. The probability of declaring a point as an outlier when in fact it is not, is less than α_{Chi}^{Outl}.

The relevant attributes of a cluster include the attributes of the intervals that make up the cluster core based on which this cluster has been computed. However, some attributes may be considered uniform, even if they are relevant for several clusters. This is the case when clusters exist in such a way that their projections on a specific attribute have equal support, and thus these projections form a uniform histogram. In such cases, it may still be possible to recover incomplete cluster cores of the involved clusters, which can be later refined, unless clusters exist in such a way that all their relevant attributes look uniform. To cover these rare cases too, we test, for each cluster, using the Chi-square test, as in Section 3.3, whether its members are uniformly distributed in the attributes initially deemed uniform. When the test indicates a non-uniform distribution, then those attributes are included in the attributes considered relevant for the cluster.

If desired, the projections of clusters on their relevant attributes can be refined as the smallest interval that the cluster members project onto.

3.6 A Note on Parameters

The performance of P3C depends on three parameters: α_{Chi}^{Unif}, α_{Binom} and α_{Chi}^{Outl}. All these parameters represent significance levels used in statistical tests. These parameters are different from typical parameters used by previous work (such as the number of clusters) in that they require no prior knowledge about the data. These parameters represent the error probability that the user is willing to accept.

Given a certain value for a significance level α, the rate of false positives reported by the statistical test is α; however, the actual number of false positives reported by the statistical test is α times the number of statistical tests that are conducted. If many statistical tests are conducted, the value of α should be adjusted in such a way that the number of false positives reported by the statistical test is acceptable by the user.

In our case, we conduct at most $d * (no\_bins - 2)$ statistical tests at significance level α_{Chi}^{Unif}; n statistical tests at significance level α_{Chi}^{Outl}; and, for each projected cluster of dimensionality p, $2 * choose(p,2) + 3 * choose(p,3) + \ldots + p * choose(p,p) + NI_p$ statistical tests at significance level α_{Binom}, where NI_p is the number of intervals in $\mathcal{I}$ that exist on different attributes than the relevant attributes of the projected cluster, and $choose(p,k)$ is the binomial coefficient: $choose(p,k) = \frac{p!}{k! * (p-k)!}$.

Parameters α_{Chi}^{Unif} and α_{Chi}^{Outl} are not as critical as α_{Binom}, and their effect can be easily understood. A false positive caused by parameter α_{Chi}^{Unif} is an interval on an attribute that has in fact uniform distribution. Such an interval is unlikely to combine with other intervals and participate in a cluster core. A false positive caused by parameter α_{Chi}^{Outl} is a data point that is incorrectly reported as an outlier. On the other hand, parameter α_{Binom} controls the aggregation of 1D intervals into cluster cores, and we cannot easily predict the effect of a false positive with respect to this parameter in the aggregation of the cluster cores.

Consequently, we fix the values of α_{Chi}^{Unif} and α_{Chi}^{Outl} to 0.001 - a standard value in statistical hypothesis testing, and let α_{Binom} be the only parameter of P3C. The robustness of P3C to α_{Binom} is studied empirically in Section 3.8.

3.7 Theoretical Complexity

Computing bins and their support on all attributes has $O(n * d)$ complexity. Marking bins on an attribute has complexity $O(no\_bins)$. Computing intervals on an attribute has complexity $O(no\_bins)$. The cluster cores computation has complexity $O(2^p)$, where p is the largest dimensionality of a projected cluster. Given a cluster (core), the complexity of computing its covariance matrix is $O(d'^2)$, where d' is the reduced dimensionality in which the refinement is performed, and the complexity of computing the Mahalanobis distance between a data point and a cluster mean is $O(d'^3)$, due to the factorization of the covariance matrix into eigenvectors and eigenvalues. Assigning $no\_unassigned\_points$ data points that do not belong to any cluster core to cluster cores based on Mahalanobis distances has complexity $O(no\_unassigned\_points * K * d'^3)$. The complexity of the EM algorithm is driven by the number of iterations until convergence; typically, this number is small. The complexity of the E-step is $O(n * K * d'^3)$ and the complexity of the M-step is $O(K * d'^2)$. Outlier detection has complexity $O(n)$. The refinement of relevant attributes for a cluster with n_c points has complexity $O(n_c * (d - d') + (d - d') * \lfloor 1 + log_2(n_c) \rfloor)$.

3.8 Experimental Evaluation

The experiments reported in this section were conducted on a Linux machine with 3 GHz CPU and 2 GB RAM.

In the following, we use "P3C_hard" to refer to the variant of P3C that computes disjoint clusters, and "P3C_soft" to refer to the variant of P3C that computes overlapping clusters. We use the term P3C when we refer collectively to both variants.

3.8.1 Compared Techniques

We compare P3C against several state-of-the-art projected and subspace clustering techniques.

As techniques representative for the family of projected clustering, we select the partitional techniques PROCLUS, SSPC, and ORCLUS, the hierarchical technique HARP, and the density-based approach MINECLUS.

FINDIT is not selected because it uses similar principles to PROCLUS, but it is based on numerous parameters and it has a large execution time. PreDeCon is not selected because our preliminary experiments indicated that PreDeCon is very sensitive to the numerous parameters required, and often it is not able to detect any clusters. DOC and FASTDOC are not selected because we select MINECLUS which is the latest improvement over DOC and FASTDOC. We intended to select PRIM, but its scalability with respect to the number of data points makes it infeasible on our synthetic data sets. We intended to compare with EPCH, but after consulting with its authors, and using the original implementation, we could not find a parameter setting that produces results with reasonable accuracy on our synthetic data sets. We experimented with FIRES, but its original implementation had a storage complexity problem.

For subspace clustering, we select MAFIA that was shown to outperform CLIQUE [54] and it is representative for techniques that are based on an Apriori-like scheme (i.e., nCluster, ENCLUS, SCHISM, DUSC, DiSH). SUBCLU is not included because its original implementation had infeasible scalability with respect to the number of data points and attributes.

The list of compared techniques is therefore as follows: P3C_hard, P3C_soft, PROCLUS, SSPC, HARP, MINECLUS, MAFIA, and ORCLUS.

3.8.2 Synthetic Data

Synthetic data sets were generated as described in [5], [79]. We study the performance of the compared techniques according to the following criteria:

1. The distribution of cluster points in the relevant subspace: 1) uniform or 2) Gaussian.
2. The number of relevant attributes that clusters can have: 1) an equal or 2) a different number of relevant attributes.
3. The average number of relevant attributes.

By combining the first 2 criteria, we obtain 4 categories of synthetic data sets: *Uniform_Equal*, *Uniform_Different*, *Gaussian_Equal*, and *Gaussian_Different*. A data set in the category *Uniform_Equal* is a data set where the cluster points are *uniformly* distributed in their relevant subspace, and clusters have an *equal* number of relevant attributes. For each category, we study the effect of the 3^{rd} criterion in data generation over the performance of the compared techniques. For this purpose, in each category, we generate data sets with $n = 10000$ data points, $d = 100$ attributes, $K = 5$ clusters (clusters sizes are 2000, 2000, 2000, 2000, and 1500 points), $5\% * n = 500$ uniformly distributed noise points, and the average number of relevant attributes in $\{2, 4, 6, 8, 10, 15, 20\}$. The clusters have axis-parallel orientation, i.e., when the cluster points are Gaussian distributed in their relevant subspace, the Gaussian distributions have diagonal covariance matrices, and when the cluster points are uniform distributed in their relevant subspace, the clusters are axis-parallel hyper-rectangles. Cluster points are uniformly distributed in $[0, 1]$ on the irrelevant attributes. The extent of clusters in their relevant attributes is between 1% and 10% of the attribute range. Various amounts of overlap were introduced among the projections of projected clusters.

3.8.3 Real Data

We tested the compared techniques on the following data sets from the UCI machine learning repository [67]: the *E.coli* data set that measures 7 predictive attributes for 336 proteins classified into 8 classes; and the *Glass* data set that measures 9 predictive attributes for 214 types of glass classified into 6 classes.

3.8.4 Experimental Setup

MINECLUS, HARP and SSPC are all tested with the original implementations. PROCLUS and ORCLUS are provided by the Biosphere project [78]. We implemented MAFIA ourselves.

On synthetic data, we set the target number of clusters to the number of implanted clusters for PROCLUS, SSPC, HARP, MINECLUS, and ORCLUS. PROCLUS and ORCLUS require the average cluster dimensionality, which is set to the known average cluster dimensionality. HARP requires the maximum percentage of outliers, which is set to the known percentage of outliers. For techniques that require other parameter settings, we set these parameters as recommended by their authors: for PROCLUS: $A = 20$, $B = 5$; for SSPC: $m = 0.5$; for MINECLUS: $w = 0.3$, $\alpha = 0.1$, $\beta = 0.25$, $maxout = 20$; for MAFIA: $\alpha = 1.5$, $\beta = 0.35$; $no\_tiny\_bins = 50$, $no\_intervals\_unif\_distrib = 5$; for ORCLUS: $\alpha = 0.5$, $k_0 = 30$, $srr = 10$.

SSPC is run without the supervision option. Except HARP, MAFIA, and P3C, all techniques are non-deterministic; thus, each of them is run 5 times, and the results are averaged. $P3C\_soft$ and MAFIA allow data points to belong to more than one cluster; the other techniques compute disjoint clusters.

For P3C, we set $\alpha_{Binom} = 1.0E - 20$. As shown in Figure 3.11, P3C is robust with respect to the value of this parameter.

On real data, we use class labels as cluster labels. We set the target number of clusters to the number of classes. For parameters such as the average cluster dimensionality, whose values are hard to determine, several values are tried and the results with best accuracy are reported.

The real data sets used were collected for classification purposes. We use such real data sets because a systematic evaluation of the compared techniques on unlabeled data is cumbersome. However, in such real data sets, most of the attributes were selected in the first place because they were considered potentially relevant for the classification problems. Consequently, the real data sets may contain only full dimensional subspace clusters or very high dimensional subspace clusters. To actually verify the capability of the competing techniques to find subspace clusters, we add 5, 10, 20, and 50 attributes, respectively, to each real data set where the data points are uniformly distributed in $[0, 1]$. Subspace clusters that may exist in the data sets, full dimensional or not, will be subspace clusters of increasingly lower dimensionality as we add more uniform attributes to the data sets.

The real data sets do not contain missing values. If missing values were present, we would have to either estimate them based on the existing data or remove some data points or some attributes that contain the missing values from the data before the clustering analysis takes place.

3.8.5 Performance Measures

We use an F value to measure the clustering accuracy. We refer to implanted clusters as *input* clusters, and to found clusters as *output* clusters. For each output cluster i, we determine the input cluster j^i with which it shares the largest number of data points. The *precision* of output cluster i is defined as the number of data points common to i and j^i divided by the total number of data points in i. The *recall* of output cluster i is defined as the number of data points common to i and j^i divided by the total number of data points in j^i. The F value of output cluster i is the harmonic mean of its precision and recall. The F value of a clustering solution is obtained by averaging the F values of all its output clusters. Similarly, we use an F value to measure the accuracy of found relevant attributes based on the matching between output and input clusters (except for ORCLUS, since it generates general sets of orthogonal vectors).

3.8.6 Accuracy Results

On synthetic data, in all performed experiments, the number of clusters discovered by P3C equals the number of implanted clusters in the data.

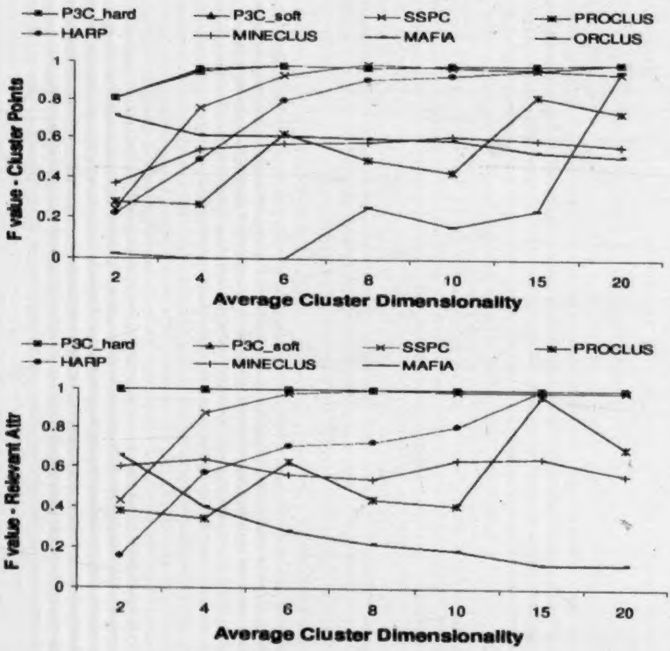

Figure 3.3: P3C and competing techniques on category Uniform_Equal

Figures 3.3, 3.4, 3.5 and 3.6 show the accuracy of the compared techniques as a function of increased average cluster dimensionality for the categories *Uniform_Equal*, *Uniform_Different*, *Gaussian_Equal*, and *Gaussian_Different*. We note that P3C_hard and P3C_soft have similar accuracies, with P3C_soft being slightly more accurate in some cases, which shows the benefit of assigning data points to more than one cluster. We observe that P3C significantly and consistently outperforms the competing techniques, both in terms of clustering accuracy and in terms of accuracy of the found relevant attributes.

The difference in performance between P3C and previous methods is particularly large for data sets that contain very low dimensional clusters embedded in high dimensional spaces. Even in these difficult cases P3C shows very high accuracies, in contrast to the modest accuracies obtained by the competing techniques. As the average cluster dimensionality increases, the accuracy of the competing techniques increases as well.

The competing techniques can be divided into two categories: techniques that are eventually able to discover the implanted clusters as the average cluster dimensionality increases, and techniques that cannot achieve that. Techniques in the first category are SSPC and HARP. Note, however, that both these techniques require as critical parameter the target number of clusters. Techniques in the second category are PROCLUS, MINECLUS, MAFIA and ORCLUS. ORCLUS has very low accuracy on data sets with low dimensional

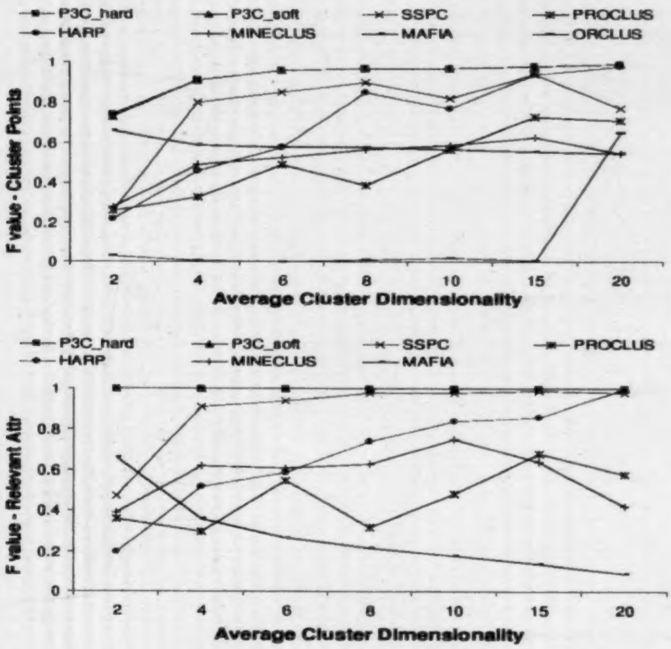

Figure 3.4: P3C and competing techniques on category Uniform_Different

clusters due to inaccurate eigen-decomposition of covariance matrices.

The accuracy results on data sets where cluster points are uniformly distributed in their relevant subspace is slightly higher than the accuracy results on data sets where cluster points are Gaussian distributed in their relevant subspace. The reason is that projections of clusters on their relevant attributes can be approximated more faithfully by the computed intervals for clusters in the former category than for clusters in the latter category.

The accuracy results on data sets where clusters have an equal number of relevant attributes are comparable with accuracy results on data sets where clusters have a different number of relevant attributes. This is to be expected, since P3C does not use in any way the average cluster dimensionality.

Accuracy results on real data sets. Figures 3.7 and 3.8 show the accuracy of the compared techniques on the E.coli and Glass data sets, as a function of increased number of uniform attributes added to the data. The first point in the graphs correspond to the original data sets with no attributes added.

We observe that P3C has higher accuracy on these data sets than the competing techniques. In addition, the accuracies of the competing techniques decrease as the number of attributes added increases, because it becomes more difficult to detect increasingly lower dimensional clusters. In contrast, the

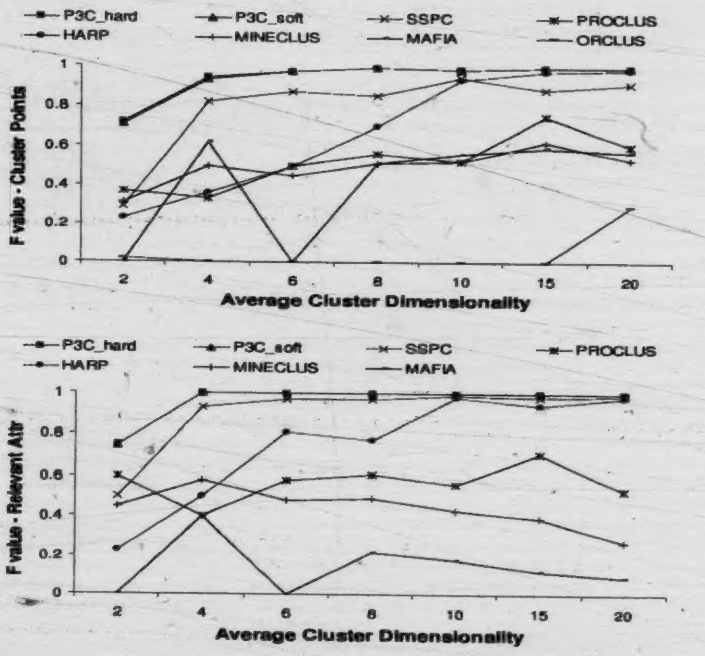

Figure 3.5: P3C and competing techniques on category Gaussian_Equal

accuracy of P3C is not affected by additional uniform attributes.

P3C_hard and P3C_soft obtain identical results. On the Ecoli data set and its extensions, P3C recovers consistently 1 2-dimensional cluster. On the Glass data set and its extensions, P3C recovers consistently 5 2-dimensional clusters.

Given a real data set with K classes, we compute the accuracy, using the F value, of a random partition of the data set into K clusters, with no outliers, so that a data point belongs to any of the clusters in the random partition with an equal probability of $1/K$. The accuracy of a random partition built in this way represents a baseline accuracy against which we can compare the accuracies of the competing techniques. On the Ecoli data set and its extensions, the accuracy of a random partition into 8 clusters is 0.19, and the accuracy of P3C is 0.61. On the Glass data set and its extensions, the accuracy of a random partition into 6 clusters is 0.22, and the accuracy of P3C is 0.55. Thus, P3C obtains better accuracy on these data sets than the accuracy of a random partition into a number of clusters that equals the number of classes.

The cluster cores computation is exponential in the largest dimensionality of a projected cluster. Thus, if, for instance, a 40-dimensional projected cluster is the projected cluster with the largest dimensionality in a data set, then the computation of cluster cores has complexity of the order 2^{40}, regardless the total number of attributes in the data. Consequently, we have encountered

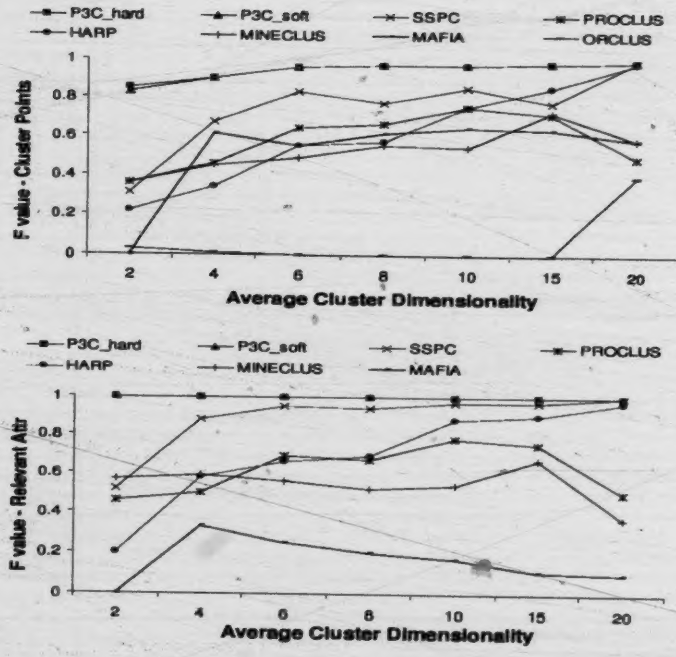

Figure 3.6: P3C and competing techniques on category Gaussian_Different

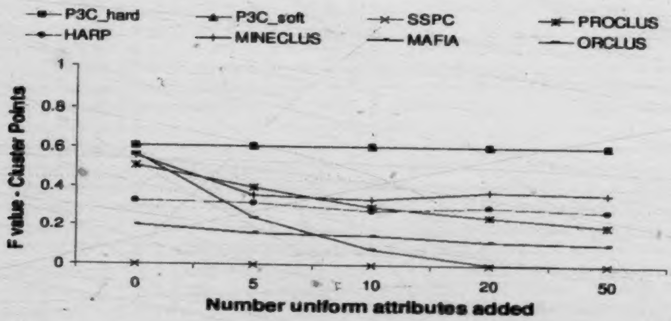

Figure 3.7: Accuracy of P3C and competing techniques on E.coli data set

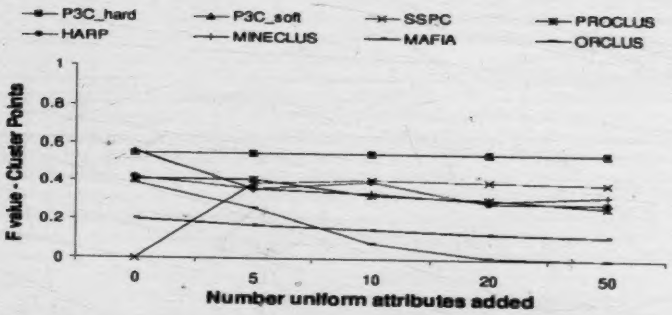

Figure 3.8: Accuracy of P3C and competing techniques on Glass data set

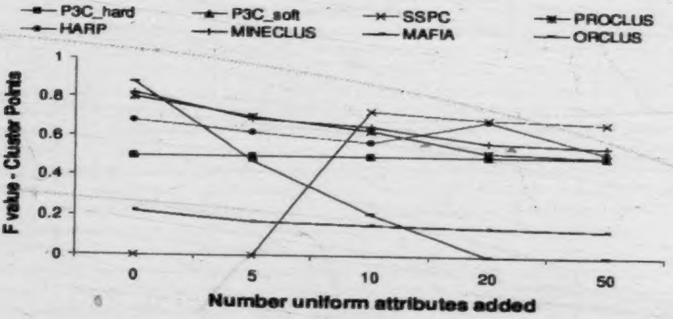

Figure 3.9: Accuracy of P3C and competing techniques on Iris data set

real data sets where the running time of P3C is large.

We note that there are real data sets where the accuracy of P3C is higher than the accuracies of some, but not all, of the compared techniques. An example is illustrated in Figure 3.9 on the *Iris* data set from the UCI machine learning repository that measures 4 predictive attributes of 150 data points classified into 3 classes. However, the accuracy of P3C is 0.5 and it is higher than the accuracy 0.34 of a random partition into 3 clusters on this data set and its extensions.

3.8.7 Robustness to Noise

To test the robustness of P3C to increasing amounts of noise in the data, we generate data sets from the category *Uniform_Equal* with $n = 10000$, $d = 100$, $K = 5$ clusters, 4 relevant attributes per cluster, and percentages of noise points as 0%, 5%, 10%, 15%, 20% and 25% of n (cluster sizes are adjusted proportionally). Figure 3.10 illustrates the result of this experiment.

The clustering accuracy of P3C decreases only slightly as more outliers are introduced. The accuracies of the competing techniques decreases as well. Even when the percentage of outliers in the data is as high as 25%, P3C still

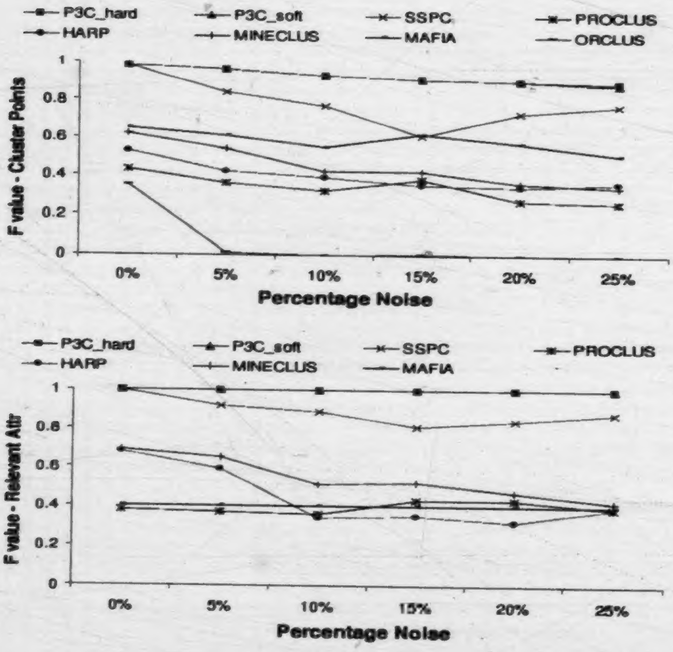

Figure 3.10: Robustness of P3C and competing techniques to noise

obtains a clustering accuracy of 89%. The accuracy of the found relevant attributes for P3C remains 100% with increasing percentages of noise.

3.8.8 Sensitivity Analysis

We have studied the sensitivity of P3C to the parameter α_{Binom}. Figure 3.11 illustrates the accuracy of P3C_soft as the parameter α_{Binom} is progressively decreased from $1.0E-10$ to $1.0E-100$ on one of our synthetic data sets. Same results are obtained for P3C_hard. We observe that P3C is robust with respect to the parameter α_{Binom}. Similar results in the sense of robustness to α_{Binom} have been obtained on all our synthetic data sets. Consequently, the parameter α_{Binom} can be set at any value in the above range.

3.8.9 Scalability Experiments

We measure the scalability of the compared techniques with respect to the database size n, database dimensionality d, and average cluster dimensionality, because these criteria have the largest impact on scalability.

In all scalability figures, the time is represented on a log scale, where the base of the log is 10. We note that the absolute running times of the compared techniques are influenced by the specific implementations of the techniques

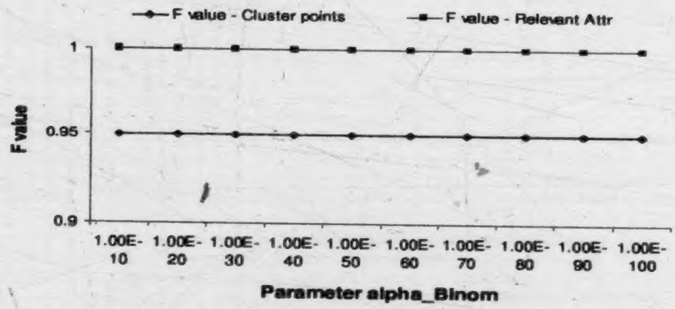

Figure 3.11: Sensitivity of P3C to parameter α_{Binom}

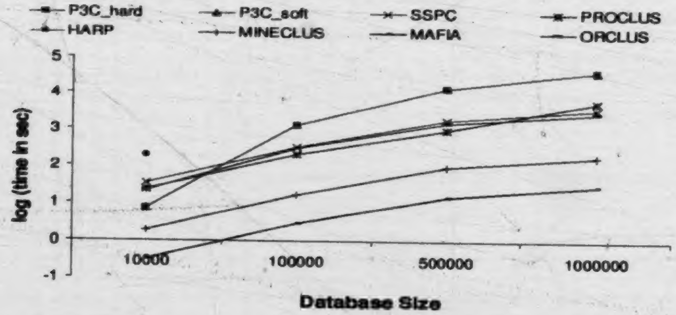

Figure 3.12: Scalability of P3C and competing techniques with increasing database size

and/or hardware used. Therefore, we are interested in the tendencies/slopes of the techniques rather than in a exact characterization of which techniques seem to be the fastest. Run times that differ by a small factor only will look similar in the log plot. However, if the run times differ by orders of magnitude, then there will be a significant difference between them in the plots, which suggests that there is a substantial difference between the techniques that cannot be overcome no matter how efficient an implementation is.

Figure 3.12 shows scalability results for increasing database sizes on synthetic data sets from category *Uniform_Equal* with $d = 10$, $K = 2$, 2 relevant attributes per cluster. HARP can only be run for the first data set with $n = 10000$. The running time of P3C increases with increasing database size because the supports of the hyper-rectangles involved in the cluster cores computation increase, and the complexity of computing the right critical value of a Binomial distribution with parameters x and *prob* increases as x increases.

Figure 3.13 shows scalability results for increasing database dimensionality on synthetic data sets from category *Uniform_Equal* with $n = 300$, $K = 2$, 2 relevant attributes per cluster. MINECLUS cannot be run for the last two data sets with $d = 500$ and $d = 1000$. The running time of P3C increases only

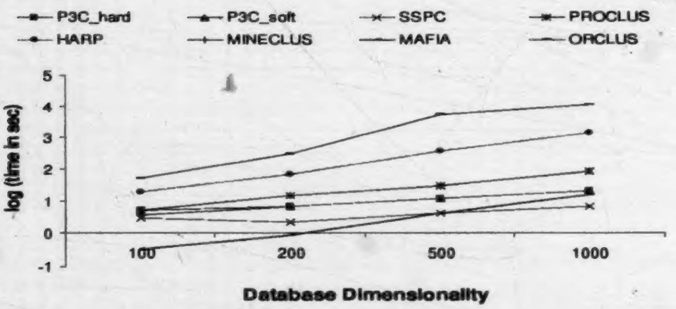

Figure 3.13: Scalability of P3C and competing techniques with increasing database dimensionality

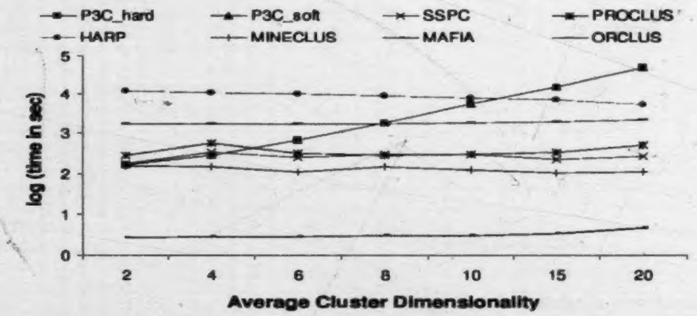

Figure 3.14: Scalability of P3C and competing techniques with increasing average cluster dimensionality

slightly with increasing data dimensionality because attributes with uniform distribution are not involved in the computation of cluster cores.

Figure 3.14 shows scalability results for increasing average cluster dimensionality on synthetic data sets from category *Uniform_Equal*. The running time of P3C increases with increasing average cluster dimensionality due to the increased complexity of cluster cores generation.

3.9 Summary

Our experimental evaluation on synthetic and real world data sets shows that P3C can effectively discover clusters in the data. P3C's accuracy can be influenced by one parameter setting. However, setting this parameter does not require critical prior knowledge about the data set. This parameter is a significance level used in a statistical test, and it represents the error probability that the user is willing to accept.

Our experiments illustrate that P3C can indeed discover low dimensional clusters embedded in high dimensional spaces. In addition, P3C consistently outperforms state of the art subspace and projected clustering techniques. P3C is also robust to noise.

P3C can compute both disjoint and overlapping clusters. This is beneficial, because some application domains may require disjoint clusters, whereas in other cases overlapping clusters may be preferable.

From an algorithmic point of view, P3C positions itself between projected and subspace clustering techniques. P3C approximates one-dimensional projections of clusters, which are subsequently aggregated in a bottom-up fashion to find the clusters. The definition of a cluster core satisfies an anti-monotonic property, which is needed for a more efficient traversal of the search space. However, unlike the other subspace clustering techniques, our criterion leverages the data model, and it is not a global density threshold.

As a drawback, P3C has exponential complexity in the dimensionality of the largest subspace where clusters exist.

Chapter 4

P3C for Categorical Data

The majority of existing subspace and projected clustering techniques are designed for *numerical* data sets, i.e., data sets where the domain of every attribute is inherently ordered. However, many real data sets are *categorical*, i.e., the attribute domains are discrete and not ordered.

Subspace and projected clustering techniques designed for numerical data are not readily applicable to categorical data sets because they may exploit geometric properties of the data space that cannot be effectively captured by categorical distance functions, use numerical computations that cannot be performed on categorical data, and/or assume an implicit order on the attributes.

Our survey of the existing subspace and projected clustering techniques for categorical data shows that they exhibit the same drawbacks as their numerical counterparts.

P3C can be extended in order to deal with categorical data. In the following, we present the extensions that need to be performed in order to make P3C applicable on categorical data.

4.1 Preliminary Definitions

We assume the same definition of a projected cluster as in the previous chapter and the same preliminary definitions as introduced in Section 3.1, but with the following additions/modifications.

For categorical data sets, we assume that each domain attribute $dom(Attr_j)$ is finite and discrete, $1 \leq j \leq d$.

For a *categorical* attribute $a \in A$, an ***interval*** I on attribute a is defined as a subset of attribute values in $dom(a)$. In this case, the width of the interval I is defined as the number of attribute values in I.

4.2 Overview of P3C for Categorical Data

P3C for categorical data differs from P3C for numerical data in two major aspects:

1. The computation of one-dimensional intervals that approximate projections of characteristic hyper-rectangles on their attributes.
2. The refinement of cluster cores into clusters and the computation of outliers.

Once 1D intervals that approximate projections of characteristic hyper-rectangles on their attributes have been computed, P3C can aggregate these intervals into cluster cores in the same way as for numerical data (Section 3.4). In addition, the refinement of relevant attributes for each cluster can be performed in the same way as for numerical data (Section 3.5).

4.3 Approximating Projections of Characteristic Hyper-rectangles on Categorical Attributes

Since the domain of a categorical attribute is finite and discrete, we consider a bin for each single value in the domain of a categorical attribute. Same as for numerical data, we use the Chi-square goodness-of-fit test to identify attributes with uniform distribution, and for the non-uniform attributes, to mark the bins with unusual high support (Section 3.3).

On numerical attributes, projections of characteristic hyper-rectangles are approximated by intervals that are simply computed as maximal sets of consecutive marked bins. We want to compute 1D intervals that approximate projections of characteristic hyper-rectangles on their attributes for categorical attributes too. To achieve this goal, we use the natural order exhibited by numerical attributes to define an "adjacency" relationship between marked bins on the *same* attribute.

Definition 4.1 Let mb_1 and mb_2 be two marked bins on the *same numerical* attribute $Attr_j$, $1 \leq j \leq d$. mb_1 and mb_2 are called "adjacent" if they are consecutive bins on attribute $Attr_j$.

Subsequently, each numerical attribute can be represented as a graph, in which the vertices are the marked bins on that attribute, and edges exist between two vertices if the corresponding marked bins are adjacent, as defined in Definition 4.1.

Property 4.1 Let $Attr_j$ be a numerical attribute, $1 \leq j \leq d$. Let $mb_{j_1}, \ldots, mb_{j_h}$ be h marked bins on attribute $Attr_j$. Then, $mb_{j_1}, \ldots, mb_{j_h}$

are consecutive bins on attribute $Attr_j$ if and only if $mb_{j_1}, \ldots, mb_{j_h}$ form a connected component in the associated graph representation.

Proof. Let us assume that $mb_{j_1}, \ldots, mb_{j_h}$ are consecutive bins on attribute $Attr_j$. Then, by Definition 4.1, there is an edge between mb_{j_l} and $mb_{j_{l+1}}$ in the associated graph representation, $1 \le l \le (h-1)$. Therefore, there is a path between any two vertices mb_{j_l} and mb_{j_i}, $\forall l, i \in \{1, \ldots, h\}$, i.e., $mb_{j_1}, \ldots, mb_{j_h}$ form a connected component.

Conversely, let as assume that $mb_{j_1}, \ldots, mb_{j_h}$ form a connected component in the associated graph representation. By Definition 4.1, each node mb_{j_l} $1 \le l \le h$ must have degree at most 2, and there must exist 2 nodes with degree exactly 1. Starting from one of the nodes of degree 1, we can infer, based on Definition 4.1, the order of the bins $mb_{j_1}, \ldots, mb_{j_h}$ on attribute $Attr_j$. Thus, $mb_{j_1}, \ldots, mb_{j_h}$ are consecutive bins.

Consequence. On numerical attributes, intervals that approximate projections of characteristic hyper-rectangles on their attributes are equivalent to connected components in the associated graph representation.

Categorical attributes lack order, and thus the notion of "consecutive" bins is not applicable. In order to determine, for each attribute, which marked bins should be merged within the same interval, we use a similar methodology to the one described in Section 3.4 to define an adjacency relationship for categorical attributes.

Specifically, let mb_1 and mb_2 be two marked bins on two *distinct* categorical attributes $Attr_i$ and $Attr_j$ $(i \neq j)$, respectively. Let no_bins_i and no_bins_j be the number of bins on $Attr_i$, respectively $Attr_j$. We want to decide whether mb_1 and mb_2 belong to the projection of the same characteristic hyper-rectangle on these attributes.

We regard mb_1 as a one-dimensional hyper-rectangle $H_1 = mb_1$, and mb_2 as an interval on attribute $Attr_j$, where $Attr_j$ does not belong to $subspace(H_1)$. As in Section 3.4, we say that there is evidence that mb_2 approximates the same projected cluster as $H_1 = mb_1$ if, for the hyper-rectangle $H_1' = mb_1 \times mb_2$, it holds that $AS(H_1' = mb_1 \times mb_2) \ge \theta^{(2)}_{\alpha_{Binom}}$, where $\theta^{(2)}_{\alpha_{Binom}}$ is the right critical value of the Binomial distribution with parameters $AS(H_1 = mb_1)$ and $\frac{1}{no\_bins_j}$ at a significance level of α_{Binom}.

Similarly, we regard mb_2 as a one-dimensional hyper-rectangle $H_2 = mb_2$, and mb_1 as an interval on attribute $Attr_i$, where $Attr_i$ does not belong to $subspace(H_2)$. We say that there is evidence that mb_1 approximates the same projected cluster as $H_2 = mb_2$ if, for the hyper-rectangle $H_2' = H_1' = mb_2 \times mb_1$, it holds that $AS(H_2' = mb_2 \times mb_1) \ge \theta^{(1)}_{\alpha_{Binom}}$, where $\theta^{(1)}_{\alpha_{Binom}}$ is the right critical value of the Binomial distribution with parameters $AS(H_2 = mb_2)$ and $\frac{1}{no\_bins_i}$ at a significance level of α_{Binom}.

Definition 4.2 Let mb_1 and mb_2 be two marked bins on two *distinct* categorical attributes $Attr_i$ and $Attr_j$ $(i \neq j)$, respectively. Let no_bins_i and no_bins_j be the number of bins on $Attr_i$, respectively $Attr_j$. Let α_{Binom} be a

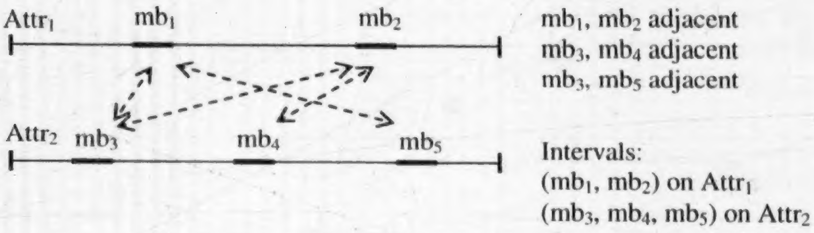

Figure 4.1: Illustration of intervals on categorical attributes

significance level. mb_1 and mb_2 belong to the projections of the same characteristic hyper-rectangle on $Attr_i$, respectively, $Attr_j$ if the following 2 conditions are satisfied:

1. $AS(mb_1 \times mb_2) \geq \theta^{(1)}_{\alpha_{Binom}}$, where $\theta^{(1)}_{\alpha_{Binom}}$ is the right critical value of the Binomial distribution with parameters $AS(mb_2)$ and $\frac{1}{no\_bins_i}$ at a significance level of α_{Binom}.
2. $AS(mb_1 \times mb_2) \geq \theta^{(2)}_{\alpha_{Binom}}$, where $\theta^{(2)}_{\alpha_{Binom}}$ is the right critical value of the Binomial distribution with parameters $AS(mb_1)$ and $\frac{1}{no\_bins_j}$ at a significance level of α_{Binom}.

Using Definition 4.2, all pairs of marked bins on distinct categorical attributes that represent the projections of the same characteristic hyper-rectangle on these attributes are computed. Subsequently, these pairs are used to define an "adjacency" relationship between marked bins on the *same categorical* attribute.

Definition 4.3 Let mb_1 and mb_2 be two marked bins on the *same categorical* attribute $Attr_i$. mb_1 and mb_2 are called "adjacent" if there is at least one marked bin mb_3 on another categorical attribute $Attr_j$, $j \neq i$, so that (mb_1, mb_3) and (mb_2, mb_3) belong to the projection of the same characteristic hyper-rectangle according to Definition 4.2.

Figure 4.1 is used to illustrate Definitions 4.2 and 4.3. In this figure, $Attr_1$ and $Attr_2$ are two categorical attributes; mb_1 and mb_2 are marked bins on attribute $Attr_1$; and mb_3, mb_4 and mb_5 are marked bins on attribute $Attr_2$. According to Definition 4.2, (mb_1, mb_3), (mb_1, mb_5), (mb_2, mb_3), and (mb_2, mb_4) belong to the projections of the same characteristic hyper-rectangle on $Attr_1$, respectively $Attr_2$. These facts are illustrated in Figure 4.1 by dotted double-arrowed lines. Thus, based on Definition 4.3, mb_1 and mb_2 on $Attr_1$ are adjacent, because there is mb_3 on $Attr_2$ so that (mb_1, mb_3) and (mb_2, mb_3) belong to the projections of the same characteristic hyper-rectangle on $Attr_1$, respectively $Attr_2$. Also based on Definition 4.3, it follows that mb_3 and mb_4 on $Attr_2$ and mb_3 and mb_5 on $Attr_2$ are adjacent.

In analogy to numerical data, intervals on categorical attributes are defined as connected components under the adjacency relationship introduced in Definition 4.3. Some subspace clustering techniques for categorical data (e.g., CACTUS, CLICKS) define intervals on categorical attributes as cliques under a different adjacency criterion. Note that cliques under the adjacency criterion in Definition 4.3 will produce overlapping intervals, which cannot be obtained on numerical data, according to the adjacency criterion in Definition 4.1. In order to have a unified treatment of numerical and categorical data, P3C computes connected components instead of cliques under its adjacency criterion.

In Figure 4.1, we obtain two intervals: interval (mb_1, mb_2) on attribute $Attr_1$, and interval (mb_3, mb_4, mb_5) on attribute $Attr_2$. With cliques instead of connected components under the adjacency criterion in Definition 4.3, we would obtain three intervals: interval (mb_1, mb_2) on attribute $Attr_1$, interval (mb_3, mb_4) on attribute $Attr_2$, and interval (mb_3, mb_5) on attribute $Attr_2$. The intervals on $Attr_2$ would overlap in mb_3. Such intervals could not have been obtained on numerical data.

4.4 Cluster Cores Refinement and Outlier Detection on Categorical Data

Let K be the number of cluster cores obtained by aggregating the intervals computed in Section 4.3 according to the computation of cluster cores described in Section 3.4. Cluster cores are axis-parallel hyper-rectangles. Similarly to the computation of clusters on numerical data, we could use some distance function for categorical data to assign the remaining data points to cluster cores. In the case of numerical data, the resulting fuzzy partitioning of the data points into K clusters is used to initialize EM for a mixture of Gaussian distributions. Based on the current definition of clusters, EM computes cluster means and covariance matrices. Such computations are not possible for categorical data; thus EM for a mixture of Gaussian distributions is not applicable.

Instead, we compute the clusters by measuring how "relevant" the bins that appear in the signatures of cluster cores are with respect to the cluster cores.

We consider the space given by the union of all ('attribute_id', 'bin_id') pairs, where 'attribute_id' is an attribute of an interval I that appears in the definition of a cluster core, and 'bin_id' is a bin that appears in the interval I on 'attribute_id'.

Each cluster core CC is represented as a vector in this space, and each entry in this vector represents the "relevance score" of a certain ('attribute_id', 'bin_id') pair with respect to the cluster core CC. Similarly to the standard "TF-IDF" scheme used in information retrieval, the relevance score of a pair (‘attribute_id’, ‘bin_id’) with respect to a cluster core CC consists of the product of two terms. The first term of the product is the fraction of cluster core points that belongs to the support set of ‘bin_id’ on attribute ‘attribute_id’. The second term of the product is the inverse of the fraction of data points that belongs to the support set of ‘bin_id’ on attribute ‘attribute_id’. In other words, the relevance score of a pair (‘attribute_id’, ‘bin_id’) with respect to a cluster core CC is proportional to the frequency at which the pair (‘attribute_id’, ‘bin_id’) appears among the points of the cluster core CC, and inverse proportional to the frequency at which the pair (‘attribute_id’, ‘bin_id’) appears among all data points.

Each data point is also represented as a vector in the aforementioned space. In this case, the entry corresponding to a (‘attribute_id’, ‘bin_id’) pair is either 1 or 0, depending on whether the data point belongs to the support set of ‘bin_id’ on attribute ‘attribute_id’ or not.

The similarity between a data point and a cluster core is defined as the dot product of their corresponding vectors. We can regard the resulting similarity matrix between data points and cluster cores as a matrix of membership probabilities by “min-max” normalizing each row.

Similarly to the numerical data, we can compute now disjoint or overlapping clusters. Disjoint clusters are obtained by assigning each point to the cluster core with the highest similarity. Overlapping clusters are obtained by assigning each point to all cluster cores with similarity larger than $1/K$.

Outliers are the points with similarity 0 to all cluster cores.

4.5 A Note on Parameters

The performance of P3C on categorical data depends on two parameters α_{Chi}^{Unif} and α_{Binom}, which have the same meaning as for numerical data. In the case of categorical data, α_{Binom} is involved not only in the computation of cluster cores, but also in the computation of intervals. Consequently, we fix α_{Chi}^{Unif} to 0.001, and we let α_{Binom} be the only parameter of P3C. We study the sensitivity of P3C for categorical data to α_{Binom} in Section 4.7.8.

4.6 Theoretical Complexity

The complexity of computing intervals on an attribute is different on categorical data than on numerical data. Computing all pairs of marked bins on distinct categorical attributes that represent the projections of the same characteristic hyper-rectangle on these attributes has complexity equal to the product of the number of marked bins on distinct categorical attributes. Computing the adjacency relationship on a categorical attribute between marked bins on that attribute (Definition 4.3) has complexity quadratic in the number of marked bins on that attribute. Computing connected components on an attribute under the adjacency relationship defined in Section 4.3 has complexity linear in the number of marked bins on that attribute plus the number of edges between the marked bins on that attribute under the adjacency criterion in Section 4.3.

The complexity of cluster cores refinement and outlier detection in the case of categorical data is given by the number of all unique ('attribute_id', 'bin_id') pairs that appear in the definition of any cluster core.

4.7 Experimental Evaluation

As in the case of numerical data, the experiments reported in this Section were conducted on a Linux machine with 3 GHz CPU and 2 GB RAM.

4.7.1 Compared Techniques

On categorical data, we evaluate P3C against SUBCAD and CLICKS. CACTUS is not included in the experiments, since CACTUS can mine only a limited class of subspace clusters, and CLICKS was shown to outperform CACTUS [84]. STIRR and its following improvement only compute cluster projections instead of the clusters.

The list of compared techniques is therefore as follows: P3C_hard, P3C_soft, SUBCAD and CLICKS.

4.7.2 Synthetic Data

We study the performance of the compared techniques according to the following criteria:

1. The number of relevant attributes that clusters can have: 1) an equal or 2) a different number of relevant attributes.

2. The average number of relevant attributes.

Based on the first criterion, we have 2 categories of synthetic data sets: category *Equal* and category *Different*. For each category, we study the effect of the second criterion in data generation over the performance of the compared techniques. For this purpose, in each category, we generate data sets with $n = 10000$ data points, $d = 100$ attributes, $K = 5$ clusters (clusters sizes are 2000, 2000, 2000, 2000, and 1500 points), $5\% * n = 500$ uniformly distributed noise points, and the average number of relevant attributes in $\{2, 4, 6, 8, 10, 15, 20\}$. The domain size of each attribute is 100 categories. Cluster points are uniformly distributed on the irrelevant attributes. Clusters span between 2 and 4 categories in their relevant attributes. Various amounts of overlap were introduced among the projections of projected clusters.

4.7.3 Real Data

We tested the compared techniques on the following data sets from the UCI machine learning repository [67]: the *Congressional Votes* data set that measures 16 categorical attributes for 435 congressmen classified into 2 classes; the *Post-Operative Patient* data set that measures 8 categorical attributes for 90 patients classified into 3 classes; the *Hepatitis* data set that measures 19 categorical and numerical attributes for 155 patients classified into 2 classes; the *Contraceptive Method Choice* data set that measures 9 categorical attributes for 1473 women classified into 3 classes; and the *Flags* data set that measures 30 categorical attributes for 194 country flags classified into 8 classes. The Hepatitis data set contains 5 numerical attributes, which are discretized based on medical data, as suggested in the documentation that comes with the data.

4.7.4 Experimental Setup

SUBCAD and CLICKS are tested with their original implementations.

SUBCAD requires the target number of clusters as a parameter. On synthetic data, we set the target number of clusters to the number of implanted clusters, and on real data, we set it to the number of classes. SUBCAD is a nondeterministic technique; thus, it is run 5 times, and the results are averaged. We have observed that the performance of CLICKS is very sensitive to the values of its parameters α and σ. Thus, we have run CLICKS 50 times (for each $\alpha \in \{15, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ and each $\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$), and we report the averaged results. For P3C, we set $\alpha_{Binom} = 1.0E-20$. As shown in Figure 4.10, P3C is robust with respect to the value of this parameter.

Same as for numerical data, in order to actually verify the capability of the compared techniques to find subspace clusters, we add 5, 10, 20, and 50 attributes, respectively, to each real data set where the data points are uniformly distributed. The added attributes are categorical attributes with 100 categories.

Some real data sets contain missing values. If the number of data points that contain missing values is small, we remove these data points. Otherwise, we keep all data points and we regard missing values as a category in itself (i.e., category "?").

4.7.5 Performance Measures

We measure the accuracy of the compared techniques in the same way as for numerical data using an F value (Section 3.8.5).

4.7.6 Accuracy Results

On synthetic data, in all performed experiments, the number of clusters discovered by P3C for categorical data equals the number of implanted clusters

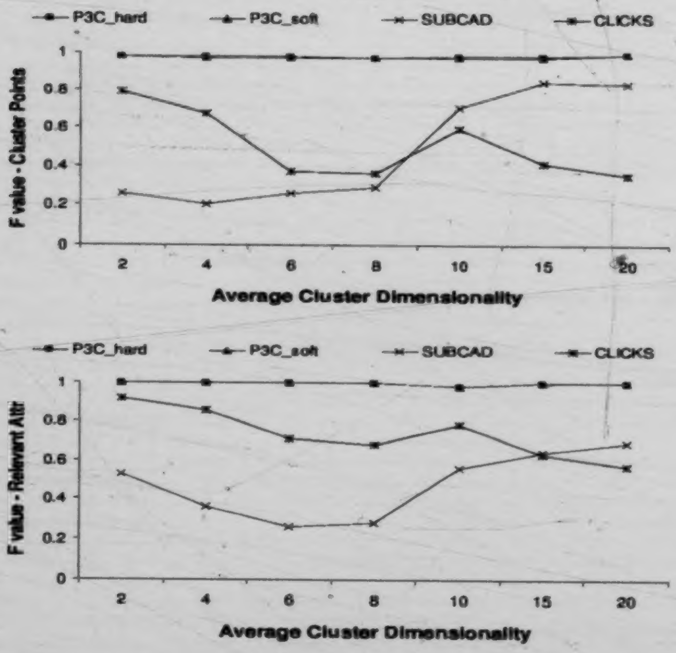

Figure 4.2: P3C and competing techniques on categorical category Equal

in the data.

Figures 4.2 and 4.3 show the accuracy of the compared techniques as a function of increased average cluster dimensionality for the categories *Equal* and *Different*.

We observe that P3C for categorical data significantly and consistently outperforms the competing techniques, both in terms of clustering accuracy and in terms of accuracy of the found relevant attributes. P3C_hard and P3C_soft for categorical data have similar accuracies.

P3C for categorical data has high accuracy even for data sets that contain very low dimensional clusters embedded in high dimensional spaces. The accuracy of SUBCAD increases with increasing average cluster dimensionality because its initialization in full dimensional space becomes more accurate as clusters become more detectable. However, SUBCAD requires the target number of clusters as a parameter and it does not compute any outliers. The accuracy of CLICKS decreases with increasing average cluster dimensionality because it computes increasingly more cliques in its graph representation, and the merging of cliques is sensitive to the parameter σ.

Similarly to numerical data, P3C for categorical data obtains comparable accuracy results on data sets where clusters have an equal number of relevant attributes versus data sets where clusters have a different number of relevant attributes.

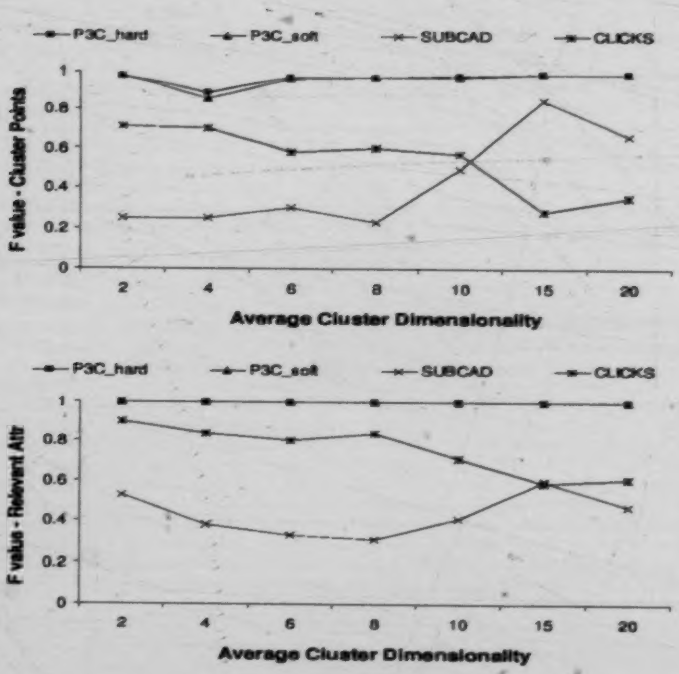

Figure 4.3: P3C and competing techniques on categorical category Different

Accuracy results on real data sets. Figures 4.4, 4.5, 4.6, 4.7 and 4.8 show the accuracy of the compared techniques on the Congressional Votes, Post-Operative Patient, Hepatitis, Contraceptive Method Choice, and respectively, Flags data sets, as a function of increased number of uniform attributes added to the data. The first point in the graphs correspond to the original data sets with no attributes added.

We observe that P3C has higher accuracy on these data sets than the competing techniques. The accuracy of P3C is not affected by additional uniform attributes. The accuracy of CLICKS decreases fast as uniform attributes are added. The accuracy of SUBCAD stays relatively constant, but it has a lower value.

P3C_hard and P3C_soft obtain identical results. On the Congressional Votes data set and its extensions, P3C recovers consistently 1 6-dimensional cluster. On the Post-Operative Patient data set and its extensions, P3C recovers consistently 1 2-dimensional cluster. On the Hepatitis data sets and its extensions, P3C recovers consistently 2 clusters, 1 2-dimensional cluster and 1 3-dimensional cluster. On the Contraceptive Method Choice data set and its extensions, P3C recovers consistently 1 8-dimensional cluster. On the Flags data set and its extensions, P3C recovers consistently 2 clusters, 1 4-dimensional cluster and 1 9-dimensional cluster.

The accuracy of a random partition into a number of clusters that equals

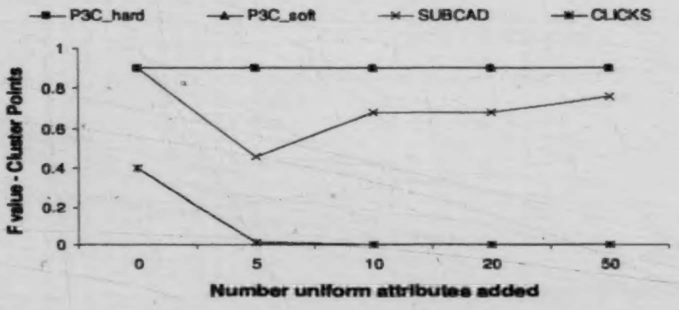

Figure 4.4: Accuracy of P3C and competing techniques on Congressional Votes data set

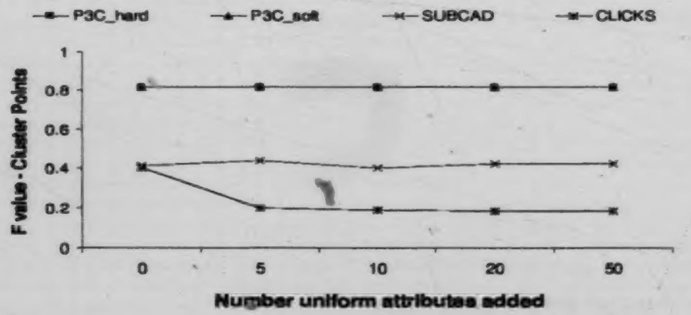

Figure 4.5: Accuracy of P3C and competing techniques on Post-Operative Patient data set

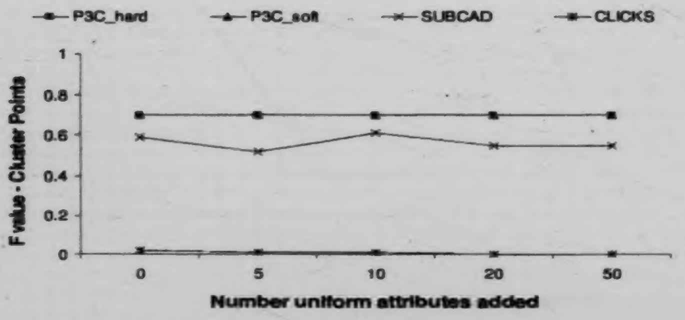

Figure 4.6: Accuracy of P3C and competing techniques on Hepatitis data set

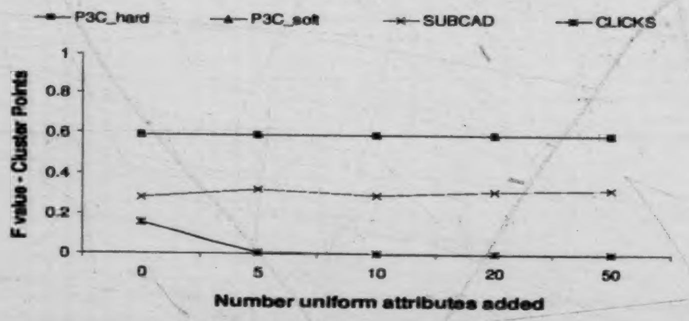

Figure 4.7: Accuracy of P3C and competing techniques on Contraceptive Method Choice data set

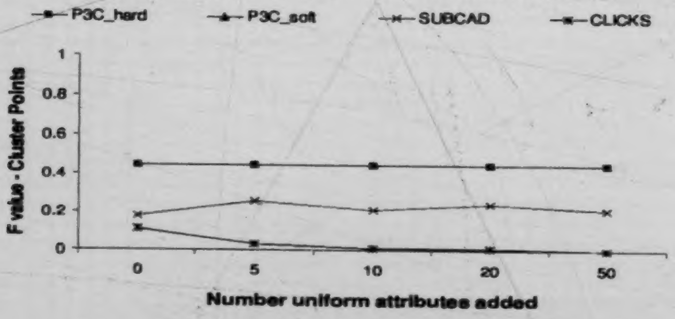

Figure 4.8: Accuracy of P3C and competing techniques on Flags data set

the number of classes on these data sets is: 0.55 for the Congressional Votes data set and its extensions; 0.45 for the Post-Operative Patient data set and its extensions; 0.61 for the Hepatitis data set and its extensions; 0.37 for the Contraceptive Method Choice data set and its extensions; and 0.17 for the Flags data set and its extensions. The accuracy of P3C on these data sets is: 0.9 for the Congressional Votes data set and its extensions; 0.82 for the Post-Operative Patient data set and its extensions; 0.7 for the Hepatitis data set and its extensions; 0.59 for the Contraceptive Method Choice data set and its extensions; and 0.45 for the Flags data set and its extensions. In all cases, the accuracy of P3C is higher than the accuracy of a random partition into a number of clusters that equals the number of classes on these data sets.

As the in case of numerical data sets, there are categorical real data sets where the running time of P3C is large due to the complexity of cluster cores generation, or categorical real data sets where the accuracy of P3C is comparable to the accuracies of the competing techniques.

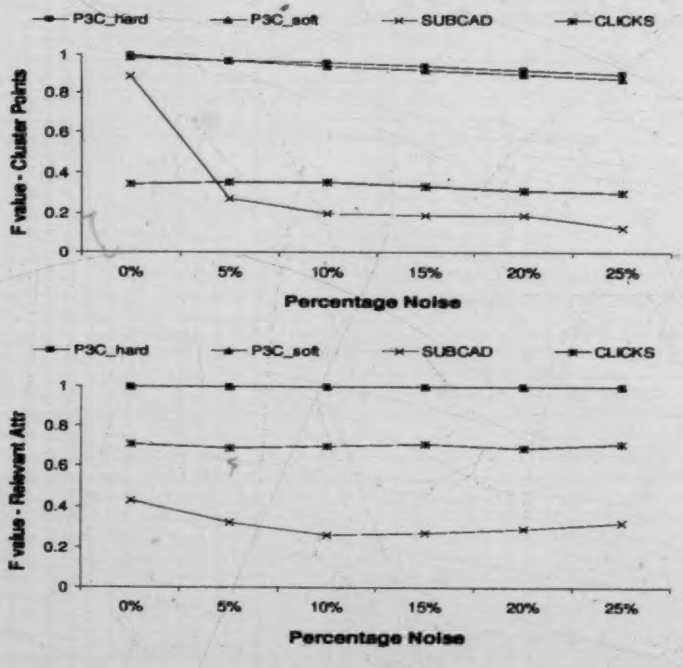

Figure 4.9: Robustness of P3C and competing techniques to noise on categorical data

4.7.7 Robustness to Noise

To test the robustness of P3C for categorical data to increasing amounts of noise in the data, we generate data sets from the category *Equal* with $n = 10000$, $d = 100$, $K = 5$ clusters, 8 relevant attributes per cluster, domain size = 100 categories, and percentages of noise points as 0%, 5%, 10%, 15%, 20% and 25% of n (cluster sizes are adjusted proportionally). Figure 4.9 illustrates the result of this experiment.

The clustering accuracy of P3C decreases only slightly as more outliers are introduced. Even when the percentage of outliers in the data is as high as 25%, P3C for categorical data still obtains a clustering accuracy of 90%. The accuracy of the found relevant attributes for P3C remains 100% with increasing percentages of noise. The accuracy of SUBCAD decreases substantially due to the fact that SUBCAD does not compute outliers. The accuracy of CLICKS is approximately constant, but it has a low value.

4.7.8 Sensitivity Analysis

We have studied the sensitivity of P3C for categorical data to the parameter α_{Binom}. Figure 4.10 illustrates the accuracy of P3C_soft as the parameter α_{Binom} is progressively decreased from $1.0E-10$ to $1.0E-100$ on one of our

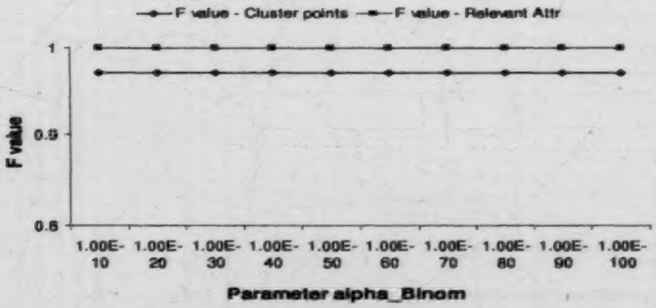

Figure 4.10: Categorical data: sensitivity of P3C to parameter α_{Binom}

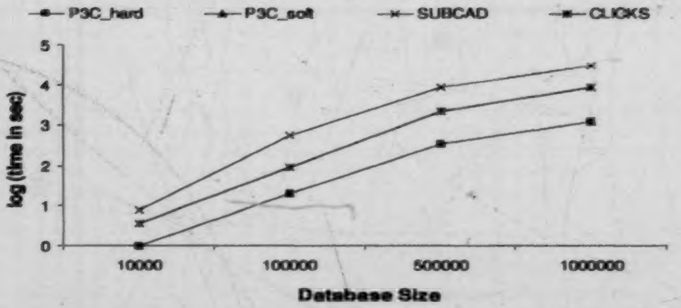

Figure 4.11: Categorical data: scalability of P3C and competing techniques with increasing database size

categorical synthetic data sets. Same results are obtained for P3C_hard. We observe that P3C is robust with respect to the parameter α_{Binom}. Similar results in terms of robustness have been obtained on all our categorical synthetic data sets. Consequently, the parameter α_{Binom} can be set at any value in the above range.

4.7.9 Scalability Experiments

We measure the scalability of the compared techniques with respect to the database size n, database dimensionality d, average cluster dimensionality, and domain size per attribute.

In all scalability figures, the time is represented on a log10 scale.

Figure 4.11 shows scalability results for increasing database sizes on categorical synthetic data sets from category *Equal* with $d = 10$, $K = 2$, 2 relevant attributes per cluster. Similarly to numerical data, the running time of P3C for categorical data increases with increasing database size due to increased complexity in the computation of the right critical values of Binomial distributions.

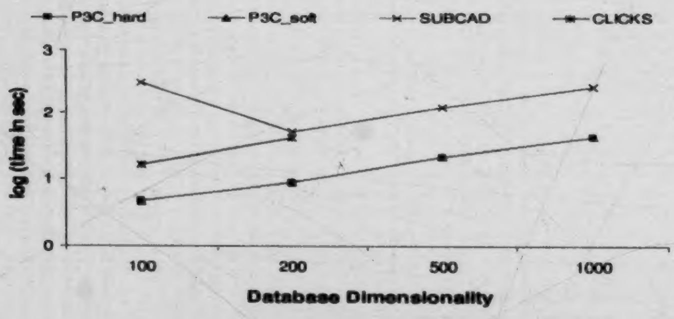

Figure 4.12: Categorical data: scalability of P3C and competing techniques with increasing database dimensionality

Figure 4.12 shows scalability results for increasing database dimensionality on categorical synthetic data sets from category *Equal* with $n = 10000$, $K = 2$, 2 relevant attributes per cluster. CLICKS cannot be run for the last two data sets with $d = 500$ and $d = 1000$. The running time of P3C increases with increasing database dimensionality. This increase is more pronounced for categorical data than for numerical data because of increased complexity in connected components generation in the computation of intervals.

Figure 4.13 shows scalability results for increasing average cluster dimensionality on categorical synthetic data sets from category *Equal*. Similarly to numerical data, the running time of P3C for categorical data increases with increasing average cluster dimensionality due to the increased complexity of cluster cores generation.

Figure 4.14 shows scalability results for increasing domain sizes on categorical synthetic data sets from category *Equal* with $n = 10000$, $d = 100$, $K = 2$, 2 relevant attributes per cluster. The running time of P3C for categorical data is unaffected by increasing domain size. The running time of CLICKS increases when the domain size per attribute is larger than 80 categories, and the running time of SUBCAD alternates, depending on how fast its objective function is minimized.

4.8 Summary

P3C for categorical data exhibits the same properties as P3C for numerical data, as discussed in Section 3.9.

P3C can deal with data sets with numerical attributes only, or with data sets with categorical attributes only. In the case of mixed data sets, the computation of intervals can be performed according to the attribute type, as discussed in Sections 3.3 and 4.3, followed by the cluster cores computation, as in Section 3.4. However, the projected clusters computation and outlier

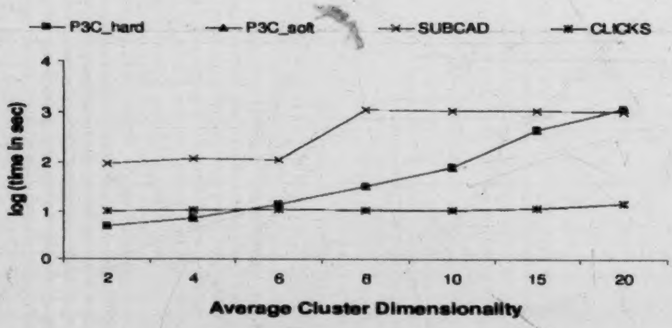

Figure 4.13: Categorical data: scalability of P3C and competing techniques with increasing average cluster dimensionality

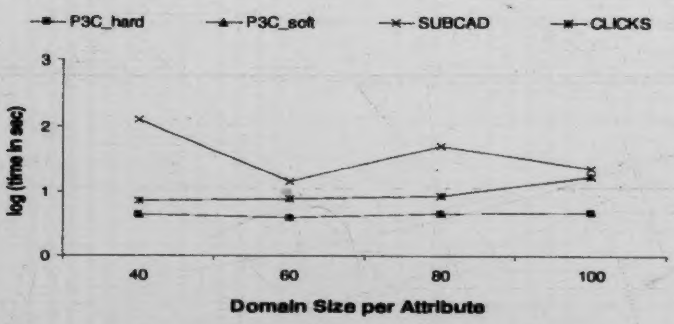

Figure 4.14: Categorical data: scalability of P3C and competing techniques with increasing domain size per attribute

detection proposed for numerical data are not applicable on the categorical attributes, since they require numerical computations that are not well defined for categorical attributes. In this case, we regard the numerical attributes as being discretized either as suggested in Section 3.3, or based on existing domain knowledge, and we apply the projected clusters computation and outlier detection for categorical data.

Chapter 5

Finding Non-Redundant, Statistically Significant Regions in High Dimensional Data

As noted in the previous chapters, subspace clustering techniques are all based on an Apriori-like scheme that involves a global density threshold, for which, however, no meaningful value is likely to exist because density naturally decreases with dimensionality. In the case of projected clustering techniques, once "closeness" and "non-closeness" measures for relevant, respectively irrelevant attributes have been defined, an exhaustive search method will report all projected clusters that comply with these measures. If only K projected clusters are desired, the techniques should define what the optimal set of K projected clusters is. The majority of projected clustering techniques fail to define what the optimal set of K projected clusters is, independent of the algorithm that finds them. In these cases, the "optimal" set of K projected clusters is the set of K projected clusters that the techniques build. Some projected clustering techniques use an objective function to define the optimal set of K projected clusters. However, these objective functions are restrictive and depend critically on parameters whose appropriate values are hard to determine by the users.

Based on our analysis, we argue that a first problem for both projected and subspace clustering is that their objectives are stated in a way that is not independent of the particular algorithm that is proposed to detect such clusters in the data - often leaving the practical relevance of the detected clusters unclear.

A second problem for most previous approaches is that they assume, explicitly or implicitly, that clusters have some point density controlled by user-defined parameters, and they will (in most cases) report some clusters. However, we have to judge if these clusters "stand out" in the data in some way, or, if, in fact, there are many structures alike in the data. Therefore, a density criterion for selecting clusters should be based on statistical principles.

5.1 Overview

Motivated by the aforementioned observations, we propose a novel problem formulation that aims at extracting from the data axis-parallel regions that "stand out" in a statistical sense. Intuitively, a *statistically significant* region is a region that contains significantly more points than expected. We consider the expectation under uniform distribution.

The set of statistically significant regions R that exist in a data set is typically highly redundant in the sense that regions that overlap with, contain, or are contained in other statistically significant regions may themselves be statistically significant. Therefore, we propose to represent the set R through a reduced, non-redundant set of (axis-parallel) statistically significant regions that in a statistically meaningful sense "explains" the existence of all the regions in R.

We will formalize these notions (Sections 5.2, 5.3, 5.4, 5.5) and formulate the task of finding a minimal set of statistically significant "explaining" regions as an optimization problem (Section 5.6).

5.2 Statistical Significance

We assume the same preliminary definitions as introduced in Section 3.1.

Let H be a hyper-rectangle in a subspace S. We use the methodology of statistical hypothesis testing (appendix A) to determine the probability that H contains $AS(H)$ data points under the null hypothesis that the n data points are uniformly distributed in subspace S. The distribution of the test statistic, $AS(H)$, under the null hypothesis is the Binomial distribution with parameters n and $vol(H)$ [12] [1], i.e.,

$$AS(H) \sim Binomial(n, vol(H)) \tag{5.1}$$

Definition 5.1 Let H be a hyper-rectangle in a subspace S. Let α_0 be a significance level. Let θ_{α_0} be the right critical value of the Binomial distribution with parameters n and $vol(H)$ at significance level α_0. H is a ***statistically significant*** hyper-rectangle if $AS(H) > \theta_{\alpha_0}$.

A statistically significant hyper-rectangle contains significantly *more* points than what is expected under uniform distribution, i.e., the probability that H contains $AS(H)$ data points when the n data points are uniformly distributed in subspace S is less than α_0 [2].

Let α_0 be an initial significance level. A value of $\alpha_0 = 0.001$ is quite common in statistical tests when a single and typically well-conceived hypothesis

---

[1] Note that if the attributes are not normalized to $[0, 1]$, we have to replace $vol(H)$ by $vol(H)/vol(S)$.

[2] Note that a two-sided test could be used to identify hyper-rectangles with significantly *less* points than expected under uniform distribution.

(i.e., one that has a high chance of being true) is tested; however, the value should be much smaller if the number of possible tests is very large, and we are actually searching for hypotheses that will pass a test; otherwise, a considerable number of false positives will be eventually reported. We will test for statistical significance hyper-rectangles in each subspace of the data set. Thus, the number of false positives increases proportionally to the number of subspaces tested. We can use a conservative Bonferroni approach and adjust the significance level α_0 for testing hyper-rectangles in a subspace of dimensionality p by the total number of subspaces of dimensionality p as $\alpha = \frac{\alpha_0}{choose(d,p)}$, where $choose(d,p)$ is the binomial coefficient, or we can use the False Discovery Rate (FDR) method [15].

Property 5.1 Statistical significance is not an anti-monotonic property.

Proof. Let $H = I_1 \times I_2$ be a hyper-rectangle in a 2-dimensional subspace, so that $width(I_1) = width(I_2) = 0.2$. Then, $vol(H) = width(I_1) * width(I_2) = 0.04$. Let $n = 200$ be the total number of data points. Let $\alpha_0 = 1.0E - 10$.

Under the null hypothesis that the n data points are uniformly distributed in $subspace(H)$, $AS(H)$ is a Binomial distributed variable with parameters $n = 200$ and $vol(H) = 0.04$. The right critical value of this Binomial distribution at significance level α_0 is $\theta^1_{\alpha_0} = 31$. Thus, according to Definition 5.1, H is a statistically significant hyper-rectangle if it contains more than 31 data points, i.e., if $AS(H) > 31$.

$H_1 = I_1$ is a one-dimensional hyper-rectangle, and, under the null hypothesis that the n data points are uniformly distributed in $subspace(H_1)$, $AS(H_1)$ is a Binomial distributed variable with parameters $n = 200$ and $vol(H_1) = width(I_1) = 0.2$. The right critical value of this Binomial distribution at significance level α_0 is $\theta^2_{\alpha_0} = 79$. Thus, according to Definition 5.1, H_1 is a statistically significant hyper-rectangle if it contains more than 79 data points, i.e., if $AS(H_1) > 79$.

If statistical significance were an anti-monotonic property, then, when $H = I_1 \times I_2$ is a statistically significant hyper-rectangle, it is guaranteed that also $H_1 = I_1$ and $H_2 = I_2$ are statistically significant hyper-rectangles. However, our example illustrates that this is not always the case. We can construct a scenario in which H contains, for instance, 45 data points - thus, H is a statistically significant hyper-rectangle - and H_1 contains, for instance, 70 data points - thus, H_1 is not a statistically significant hyper-rectangle.

Since statistical significance is *not* anti-monotonic, Apriori-like bottom-up construction of statistically significant hyper-rectangles is not possible.

5.3 Relevant vs. Irrelevant Attributes

In the following, we discuss two interesting properties of statistical significance.

Let H be a hyper-rectangle in a subspace S. As the dimensionality of S

increases, $vol(H)$ decreases towards 0 [3]. Consequently, for a given significance level α_0, the critical value θ_{α_0} in Definition 5.1 decreases towards 1. Thus, in high dimensional subspaces, hyper-rectangles H with few points may be statistically significant.

A second interesting aspect of statistical significance is given next.

Property 5.2 Let α_0 be a significance level. Let H be a statistically significant hyper-rectangle in a subspace S. Let a be an attribute so that $a \notin S$, where the coordinates of the points in $SuppSet(H)$ are uniformly distributed in $dom(a)$. Let $H' = H \times I'$, where $attr(I') = a$ and $SuppSet(I') = H$, i.e., $I' = [l, u]$ so that $l = min\{x.a | x \in SuppSet(H)\}$, and $u = max\{x.a | x \in SuppSet(H)\}$.

Then, hyper-rectangle H' is statistically significant in subspace $S' = S \cup \{a\}$.

Proof. Let X, X' be two Binomial distributed variables so that:

$$X \sim Binomial(n, vol(H)) \tag{5.2}$$

$$X' \sim Binomial(n, vol(H')) \tag{5.3}$$

For simplicity of notation, let A and B be the right critical values of the distributions $Binomial(n, vol(H))$ and $Binomial(n, vol(H'))$, respectively, at significance level α_0. By the definition of a Binomial distribution and Equation (A.1):

$$A, B \in \{0, 1, \ldots, n\} \tag{5.4}$$

$$\alpha_0 = Probability(X \geq A) \tag{5.5}$$

$$\alpha_0 = Probability(X' \geq B) \tag{5.6}$$

Since H is a statistically significant hyper-rectangle, by definition 5.1:

$$AS(H) > A \tag{5.7}$$

By the construction of H', it holds that:

$$AS(H) = AS(H') \tag{5.8}$$

$$vol(H') \leq vol(H) \tag{5.9}$$

Since X is a Binomial distributed variable as in (5.2), $Probability(X \geq l)$, $l \in \{0, 1, \ldots, n\}$ is defined as the probability of obtaining l or more successes in n independent "yes/no" experiments, where each experiment has the probability of success $vol(H)$. $Probability(X' \geq l)$, $l \in \{0, 1, \ldots, n\}$, is similarly defined.

---

[3] Note that if the attributes are not normalized to $[0, 1]$, then $vol(H)/vol(S)$ decreases towards 0.

Because of (5.9), it holds that [28]:

$$Probability(X \geq l) \geq Probability(X' \geq l), \forall l \in \{0, 1, \ldots, n\} \tag{5.10}$$

Therefore, based on (5.4) and (5.10):

$$Probability(X \geq A) \geq Probability(X' \geq A). \tag{5.11}$$

Based on (5.5) and (5.11):

$$Probability(X' \geq A) \leq \alpha_0. \tag{5.12}$$

From (5.6) and (5.12):

$$B \leq A \tag{5.13}$$

Finally, based on (5.8), (5.7), and (5.13):

$$AS(H') = AS(H) > A \geq B \tag{5.14}$$

Thus, H' is a statistically significant hyper-rectangle in $S' = S \cup \{a\}$.

Reporting statistically significant hyper-rectangles such as H' does not add any information, since their existence is "caused" by the existence of other statistically significant hyper-rectangles to which intervals have been added in which the points are uniformly distributed along the whole range of the corresponding attributes.

To deal with these aspects of statistical significance, we introduce the concept of "relevant" attributes versus "irrelevant" attributes for a hyper-rectangle H.

Definition 5.2 Let H be a hyper-rectangle in a subspace S. An attribute $a \in S$, is called ***relevant*** for H if points in $SuppSet(H)$ are *not* uniformly distributed in $dom(a)$; otherwise it is called ***irrelevant*** for H.

To test whether points in $SuppSet(H)$ are uniformly distributed in the domain of an attribute a, we use the Kolmogorov-Smirnov goodness-of-fit test for the uniform distribution [65].

The null hypothesis is that the points in $SuppSet(H)$ are uniformly distributed on the domain of an attribute a. The Kolmogorov-Smirnov test statistic computes the absolute difference between the theoretical cumulative distribution of the distribution being tested (in this case the uniform distribution on $dom(a) = [0, 1]$), and the empirical cumulative distribution of the projections of the points in $SuppSet(H)$ on attribute a. The distribution of the Kolmogorov-Smirnov test statistic under the null hypothesis is the Kolmogorov distribution [65].

Let α_K be a significance level, and let θ_{α_K} be the right critical value of the Kolmogorov distribution at significance level α_K. θ_{α_K} can be found in pre-computed tables, or it can be approximated numerically. Attributes a for which the Kolmogorov-Smirnov test statistic is larger than θ_{α_K} will be reported as non-uniform, or equivalently, relevant for H. The statistical test tells us that the probability of declaring an attribute non-uniform/relevant when in fact it is uniform/irrelevant is very small, i.e., less that α_K.

5.4 Redundancy-oblivious Problem Definition

Given a data set D of n d-dimensional points, we would like to find in each subspace all hyper-rectangles that satisfy Definitions 5.1 and 5.2. The number of hyper-rectangles in a certain subspace can be infinite. However, we consider, for each subspace, all unique Minimum Bounding Rectangles (MBRs) formed with data points instead of all possible hyper-rectangles. The reason is that adding empty space to an MBR keeps its support constant, but it increases its volume; thus, it only decreases its statistical significance.

Definition 5.3 Given a data set D of n d-dimensional points. We define a ***subspace cluster*** as an MBR formed with data points in some subspace so that the MBR 1) is statistically significant, and 2) has only relevant attributes.

Redundancy-oblivious problem definition. Find all unique subspace clusters in a set of n d-dimensional points.

For any non-trivial values of n and d, the *size of the search space* for the redundancy-oblivious problem definition is obviously very large. There are $2^d - 1$ subspaces, and the number of unique MBRs in each subspace S, that contain at least 2 points, assuming all coordinates of n points to be distinct in S, is at least $choose(n, 2)$ and upper bounded by $choose(n, 2) + choose(n, 3) + \ldots + choose(n, 2 \times dim(S))$.

The *size of the solution* to the redundancy-oblivious problem definition can be quite large as well, even if the overall distribution is generated by only a few "true" subspace clusters $\{T_1, \ldots, T_K\}$, $K \geq 1$, plus uniform background noise:

1. For each T_i, $1 \leq i \leq K$, other subspace clusters may exist around it in $subspace(T_i)$, formed by subsets of points in $SuppSet(T_i)$ plus possibly neighboring points in $subspace(T_i)$. Some of these cases are illustrated in Figures 5.1(a), 5.1(b) and 5.1(c).

2. Subspace clusters may also exist in lower or higher dimensional subspaces of $subspace(T_i)$ due to the existence of T_i. Figure 5.1(d) illustrates for a true 2-dimensional subspace cluster in the xy-plane an induced 3-dimensional subspace cluster and two 1-dimensional subspace clusters.

3. Additional subspace clusters may also exist whose points or attributes belong to different T_i. Figure 5.1(e) illustrates a subspace cluster induced by two subspace clusters from the same subspace. Figure 5.1(f)

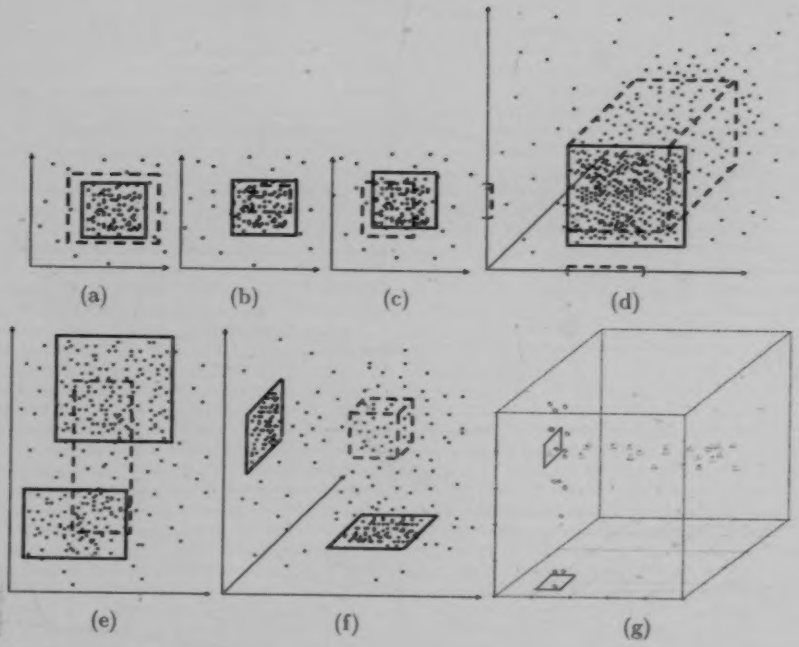

Figure 5.1: (a)-(f) Redundancy in R (solid/dotted lines for true/"induced" subspace clusters) (g) Example data

illustrates a subspace cluster induced by two subspace clusters from different subspaces.

4. Combinations of all these cases are possible as well.

The number of subspace clusters that exist only because of the "true" subspace clusters is typically very large. For instance, the total number of subspace clusters in even the simple data set depicted in Figure 5.1(g) —with 50 points and two 2-dimensional subspace clusters, which are embedded in a 3-dimensional space— is 656.

Conceptually, the solution R to the redundancy-oblivious problem definition contains three types of elements: 1) a set of subspace clusters T representing the "true" subspace clusters, 2) a set ϵ representing the false positives reported by the statistical tests, and 3) a set of subspace clusters F representing subspace clusters that exist only because of the subspace clusters in T and possibly ϵ, i.e.,

$$R = T \cup \epsilon \cup F \tag{5.15}$$

We argue that reporting the entire set R is not only computationally expensive, but it will also overwhelm the user with a highly ***redundant*** amount of information, because of the large number of elements in F.

5.5 Explain Relationship

Our goal is to represent the set R of all subspace clusters in a given data set by a reduced set P^{opt} of subspace clusters such that the existence of each subspace cluster $H \in R$ can be *explained* by the existence of the subspace clusters P^{opt}, and P^{opt} should be a smallest set of subspace clusters with that property. Ideally, $P^{opt} = T \cup \epsilon$.

To achieve this goal, we have to define an appropriate *Explain* relationship, which is based on the following intuition. We can think of the overall data distribution as being generated by the "true" subspace clusters, which we hope to capture in the set P^{opt}, plus background noise. We can say that the actual support $AS(H)$ of a subspace cluster H can be *explained* by a set of subspace clusters P, if $AS(H)$ is consistent with the assumption that the data was generated by only the subspace clusters in P and background noise.

More formally, if we have a set $P = \{P_1, ..., P_K\}$ of subspace clusters that should explain all subspace clusters in R, we assume that the overall distribution is a distribution mixture of $K+1$ components, K components corresponding to (derived from) the K elements in P and the $K+1$ component corresponding to background noise, i.e.,

$$f(x;\Theta) = \sum_{k=1}^{K+1} \mu_k f_k(x;\theta_k) \tag{5.16}$$

where θ_k are the parameters of each component density, and μ_k are the proportions of the mixture.

Conceptually, to justify that an observed subspace cluster H is explained by P, we have to test that the actual support $AS(H)$ of H is *not significantly* larger or smaller than what can be expected under the given model. Again, this can in theory be done using a statistical test, if we can determine left and right critical values for the distribution of the test statistics $AS(H)$, given a desired significance level.

Practically, there are limitations to what can be done analytically to apply such a statistical test. An analytical solution requires to first estimate the parameters and mixing proportions of the model, using the data and information that can be derived from the set P; and then, an equation for the distribution of $AS(H)$ has to be derived from the equation for the mixture model. Obviously, this is challenging (if not impossible) for more complex forms of distributions.

In the following, we show how to define the *Explain* relationship assuming that all component densities are Uniform distributions. Let the $K+1$ component be the uniform background noise in the whole space, i.e.,

$$f_{K+1}(x) \sim Uniform([0,1] \times \ldots \times [0,1]) \tag{5.17}$$

For the other components, corresponding to $P_k \in P$, we assume that data is generated such that in *subspace*(P_k), $1 \leq k \leq K$, the points are uniformly distributed in the corresponding intervals of P_k (and uniformly distributed in the whole domain in the remaining attributes, since these are the irrelevant attributes for P_k). Formally, if P_k has m_k relevant attributes, i.e., $P_k = I_1^k \times \ldots \times I_{m_k}^k$, and the d attributes are ordered as $(attr(I_1^k), \ldots, attr(I_{m_k}^k), [0,1] \ldots, [0,1])$, the k-th component density is given by

$$f_k(x) \sim Uniform(I_1^k \times \ldots \times I_{m_k}^k \times [0,1] \times \ldots \times [0,1]) \tag{5.18}$$

To determine whether the existence of a subspace cluster $H = I_1^H \times \ldots \times I_{m_H}^H$ is consistent with such a model, we have to estimate the possible contribution of each component density to H. For a component density f_k, that contribution is proportional to the volume of the intersection between f_k and H in the subspace of H, i.e., we have to determine the part of f_k that lies in H. This intersection is —like H— an m_H-dimensional hyper-rectangle $\pi_H(P_k)$ that can be computed as following. For f_k, $1 \leq k \leq K$, let $P_k = I_1^k \times \ldots \times I_{m_k}^k$, and for f_{K+1}, i.e. background noise, let $P_{K+1} = [0,1] \times \ldots \times [0,1]$:

$$\pi_H(P_k) = I_1^{\pi_H} \times \ldots \times I_{m_H}^{\pi_H}, \tag{5.19}$$

where

$$I_i^{\pi_H} = \begin{cases} I_i^H \cap I_j^k, \; if \; \exists j : attr(I_j^k) = attr(I_i^H) \\ I_i^H, \; else \end{cases}$$

Because ~~f_k is a~~ uniform distribution, the number of points in $\pi_H(P_k)$ generated by f_k follows a Binomial distribution

$$Binomial(n_k, \frac{vol(\pi_H(P_k))}{vol(P_k)}) \tag{5.20}$$

with expected value $n_k * \frac{vol(\pi_H(P_k))}{vol(P_k)}$, where n_k is the total number of points generated by f_k, and $\frac{vol(\pi_H(P_k))}{vol(P_k)}$ is the fraction of P_k that intersects H.

The numbers n_k can easily (under our assumptions) be estimated using the total number of points n and the information about the actual supports $AS(P_i)$ of the subspace clusters $P_i \in P$ in the data set. For any of the components f_i, $1 \leq i \leq K+1$, the number of points generated by that component is, according to the data model, equal to the observed number of points in P_i, $AS(P_i)$, minus the contributions n_j of the other components f_j, $j \neq i$, and $P_{K+1} = [0,1] \times \ldots \times [0,1]$ (for the background noise f_{K+1}):

$$n_i = AS(P_i) - \sum_{1 \leq j \leq K+1, j \neq i} \frac{vol(\pi_{P_i}(P_j))}{vol(P_j)} * n_j \tag{5.21}$$

where $\pi_{P_i}(P_j)$ is the intersection of hyper-rectangle P_j with hyper-rectangle P_i as defined in Equation (5.19). The equations in (5.21) can easily be solved for n_i since (5.21) is a system of $K+1$ linear equations in $K+1$ variables.

The system (5.21) is equivalent to:

$$n_{K+1} = n - (n_1 + \dots n_K)$$
$$A * [n_1, \dots, n_K]^T = b$$

where A is a $K \times K$ matrix, $A = (a_{i,j})_{1 \le i,j \le K}$, where

$$a_{i,j} = \begin{cases} 1 - vol(P_i),\ i = j \\ \frac{vol(\pi_{P_i}(P_j))}{vol(P_j)} - vol(P_i),\ i \neq j \end{cases}$$

and b is a $K \times 1$ vector $b = (b_i)_{1 \le i \le K}$ so that

$$b_i = AS(P_i) - n * vol(P_i)$$

The solution $(n_k)_{1 \le k \le K+1}$ to (5.21) consists, in general, of real numbers. Since $(n_k)_{1 \le k \le K+1}$ represent the number of data points generated by $(f_k)_{1 \le k \le K+1}$, we convert $(n_k)_{1 \le k \le K+1}$ to integers by taking the largest integer value that is not greater than n_k, for each $k \in \{1, \dots, K+1\}$.

The solution to (5.21) may include negative numbers. This indicates inconsistency with our assumptions, i.e., the data points within some $P_k \in P$ are not uniformly distributed in the corresponding intervals of P_k. The solution to (5.21) may not be unique when the system (5.21) is singular, which indicates redundancy in the set P. In these cases, we discard the set P as a possible candidate for the optimal solution P^{opt} .

We want to say that a set of subspace clusters P, plus background noise, explains a subspace cluster H if the observed number of points in H is *consistent* with this assumption and not *significantly* larger or smaller that expected. From the Binomial distributions (5.20), we can derive a lower and an upper bound on the number of points in H that could be generated by component density f_k, without this number being statistically significant; these are the left $\theta^L_{\alpha_0}(k)$, respectively right $\theta^R_{\alpha_0}(k)$, critical values of this Binomial distribution, with significance level α_0.

By summing up these bounds for each component density, we obtain a range $[ES^L_H, ES^U_H]$ of the number of points in H that could be accounted for just by the presence of the subspace clusters in P, plus background noise, i.e.,

$$ES^L_H = \sum_{k=1}^{K+1} \theta^L_{\alpha_0}(k) \tag{5.22}$$

$$ES^U_H = \sum_{k=1}^{K+1} \theta^R_{\alpha_0}(k) \tag{5.23}$$

If $AS(H)$ falls into this range, we can say that $AS(H)$ is consistent with P, or that P is in fact sufficient to explain the observed number of points in H.

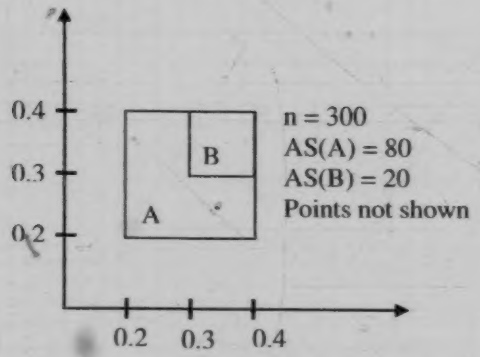

Figure 5.2: Illustration of $\{A\}$ explains B, $\{B\}$ does not explain A

Definition 5.4 Let $P \cup \{H\}$ be a set of subspace clusters. P *explains* H if and only if $AS(H) \in [ES_H^L, ES_H^U]$.

Property 5.3 Let $\{H\} \cup P$ be a set of subspace clusters, H not in P. Then, $\{H\} \cup P$ explains H.

Proof. Since H is part of the explaining set $\{H\} \cup P$, from (5.21), it follows that:

$$AS(H) = n_H + \sum_{1 \le k \le K+1} \frac{vol(\pi_H(P_k))}{vol(P_k)} * n_k \tag{5.24}$$

Thus, $AS(H)$ is the sum of the expected values of the Binomial distributions given in (5.20), $\forall k \in \{1, \ldots, K+1\}$, plus the expected value of the Binomial distribution $Binomial(n_H, \frac{vol(\pi_H(H))}{vol(H)} = 1)$, which represents the component H. Since the expected value of a Binomial distribution is between the left and right critical values of the Binomial distribution, it follows that $AS(H) \in [ES_H^L, ES_H^U]$, i.e., $\{H\} \cup P$ explains H.

Consequence. The *Explain* relationship is "reflexive", i.e., $\{H\}$ explains $H, \forall H \in R$.

Property 5.4 The *Explain* relationship is not "symmetric", i.e., given two subspace clusters $A, B \in R$, $A \neq B$, it is possible that $\{A\}$ explains B, but $\{B\}$ does not explain A.

Proof. Figure 5.2 illustrates two MBRs A and B, so that $AS(A) = 80$, $vol(A) = 0.2 * 0.2 = 0.04$, $AS(B) = 20$, $vol(B) = 0.1 * 0.1 = 0.01$. The data set has $n = 300$ data points. The data points are omitted from the figure for better clarity.

Let $\alpha_0 = 1.0E-10$ be a significance level. A is a statistical significant MBR because the right critical value of a Binomial distribution with parameters $n = 300$ and $vol(A) = 0.04$ at significance level $\alpha_0 = 1.0E-10$ is 39, and $AS(A) = 80 > 39$. B is a statistical significant MBR because the right critical value of a Binomial distribution with parameters $n = 300$ and $vol(B) = 0.01$ at significance level $\alpha_0 = 1.0E-10$ is 19, and $AS(B) = 20 > 19$. We assume that the points in $SuppSet(A)$ and $SuppSet(B)$ are not uniformly distributed on the attributes depicted in Figure 5.2, and thus A and B are subspace clusters.

We show that $\{A\}$ explains B. We solve system (5.21) for A and the uniform background noise, and we obtain the solution (n_A, n_U), where n_A is the number of points generated by the component density associated with A, and n_U is the number of points generated by the component density associated with the uniform background noise. Specifically, we obtain $n_A = \frac{AS(A)-n*vol(A)}{1-vol(A)} = 70$ and $n_U = n - n_A = 230$. Consequently, the Binomial distributions (5.20) are $Binomial(n_A, \frac{vol(A \cap B)}{vol(A)}) = Binomial(70, 0.25)$ and $Binomial(n_U, vol(B)) = Binomial(230, 0.01)$. At significance level $\alpha_0 = 1.0E - 10$, the left and right critical values of the first Binomial distribution are 0 and 43, and the left and right critical values of the second Binomial distribution are 0 and 17. By summing up these bounds, we obtain the range $[0, 60]$. Because $AS(B) = 20 \in [0, 60]$, it holds that $\{A\}$ explains B.

We show that $\{B\}$ does not explain A. We solve system (5.21) for B and the uniform background noise, and we obtain the solution (n_B, n_U), where n_B is the number of points generated by the component density associated with B, and n_U is the number of points generated by the component density associated with the uniform background noise. Specifically, we obtain $n_B = \frac{AS(B)-n*vol(B)}{1-vol(B)} = 17$ and $n_U = n - n_B = 283$. Consequently, the Binomial distributions (5.20) are $Binomial(n_B, \frac{vol(B \cap A)}{vol(B)}) = Binomial(17, 1)$ and $Binomial(n_U, vol(A)) = Binomial(283, 0.04)$. At significance level $\alpha_0 = 1.0E - 10$, the left and right critical values of the first Binomial distribution are 17 and 17, and the left and right critical values of the second Binomial distribution are 0 and 37. By summing up these bounds, we obtain the range $[17, 54]$. Because $AS(A) = 80 > 54$, it follows that $\{B\}$ does not explain A.

Property 5.5 The *Explain* relationship is not "transitive", i.e., given three subspace clusters $A, B, C \in R$, $A \neq B$, $B \neq C$, $A \neq C$, it is possible that $\{A\}$ explains B, $\{B\}$ explains C, but $\{A\}$ does not explain C.

Proof. Figure 5.3 illustrates three MBRs A, B, and C, so that $AS(A) = 60$, $vol(A) = 0.2 * 0.2 = 0.04$, $AS(B) = 31$, $vol(B) = 0.1 * 0.1 = 0.01$, $AS(C) = 30$, $vol(C) = 0.05 * 0.05 = 0.0025$. The data set has $n = 300$ data points. The data points are omitted from the figure for better clarity.

Let $\alpha_0 = 1.0E-10$ be a significance level. A is a statistical significant MBR because the right critical value of a Binomial distribution with parameters $n = 300$ and $vol(A) = 0.04$ at significance level $\alpha_0 = 1.0E - 10$ is 39, and $AS(A) = 60 > 39$. B is a statistical significant MBR because the right critical value of a Binomial distribution with parameters $n = 300$ and $vol(B) = 0.01$ at significance level $\alpha_0 = 1.0E - 10$ is 19, and $AS(B) = 31 > 19$. C is a statistical significant MBR because the right critical value of a Binomial

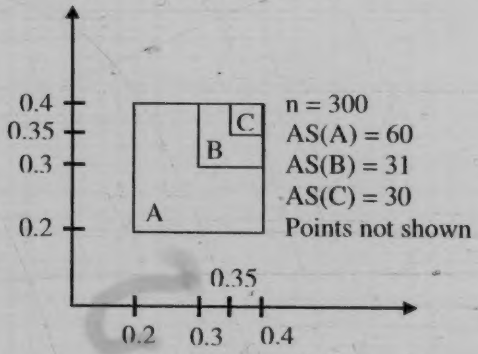

Figure 5.3: Illustration of $\{A\}$ explains B, $\{B\}$ explains C, $\{A\}$ does not explain C

distribution with parameters $n = 300$ and $vol(C) = 0.0025$ at significance level $\alpha_0 = 1.0E - 10$ is 11, and $AS(C) = 30 > 11$. We assume that the points in $SuppSet(A)$, $SuppSet(B)$, and $SuppSet(C)$ are not uniformly distributed on the attributes depicted in Figure 5.3, and thus A, B, and C are subspace clusters.

We show that $\{A\}$ explains B. We solve system (5.21) for A and the uniform background noise, and we obtain the solution (n_A, n_U), where n_A is the number of points generated by the component density associated with A, and n_U is the number of points generated by the component density associated with the uniform background noise. Specifically, we obtain $n_A = \frac{AS(A) - n * vol(A)}{1 - vol(A)} = 50$ and $n_U = n - n_A = 250$. Consequently, the Binomial distributions (5.20) are $Binomial(n_A, \frac{vol(A \cap B)}{vol(A)}) = Binomial(50, 0.25)$ and $Binomial(n_U, vol(B)) = Binomial(250, 0.01)$. At significance level $\alpha_0 = 1.0E - 10$, the left and right critical values of the first Binomial distribution are 0 and 34, and the left and right critical values of the second Binomial distribution are 0 and 18. By summing up these bounds, we obtain the range $[0, 52]$. Because $AS(B) = 31 \in [0, 52]$, it holds that $\{A\}$ explains B.

We show that $\{B\}$ explains C. We solve system (5.21) for B and the uniform background noise, and we obtain the solution (n_B, n_U), where n_B is the number of points generated by the component density associated with B, and n_U is the number of points generated by the component density associated with the uniform background noise. Specifically, we obtain $n_B = \frac{AS(B) - n * vol(B)}{1 - vol(B)} = 28$ and $n_U = n - n_B = 272$. Consequently, the Binomial distributions (5.20) are $Binomial(n_B, \frac{vol(B \cap C)}{vol(B)}) = Binomial(28, 0.25)$ and $Binomial(n_U, vol(C)) = Binomial(272, 0.0025)$. At significance level $\alpha_0 = 1.0E - 10$, the left and right critical values of the first Binomial distribution are 0 and 23, and the left and right critical values of the second Binomial distribution are 0 and 11. By summing up these bounds, we obtain the range $[0, 34]$. Because $AS(C) = 30 \in [0, 34]$, it holds that $\{B\}$ explains C.

We show that $\{A\}$ does not explain C. We solve system (5.21) for A and the uniform background noise, and we obtain the solution (n_A, n_U), where n_A is the number of points generated by the component density associated with A, and n_U is the number of points generated by the component density associated with the uniform background noise. Same as above, we obtain $n_A = 50$ and $n_U = 250$. Consequently, the Binomial distributions (5.20) are $Binomial(n_A, \frac{vol(A \cap C)}{vol(A)}) = Binomial(50, 0.0625)$ and $Binomial(n_U, vol(C)) = Binomial(250, 0.0025)$. At significance level $\alpha_0 = 1.0E - 10$, the left and right critical values of the first Binomial distribution are 0 and 18, and the left and right critical values of the second Binomial distribution are 0 and 10. By summing up these bounds, we obtain the range $[0, 28]$. Because $AS(C) = 30 > 28$, it follows that $\{A\}$ does not explain C.

Property 5.6 The *Explain* relationship is not monotonic in the following sense. Let $P \cup \{H\} \cup \{X\}$ be a set of subspace clusters, H, X not in P, $X \neq H$. It is possible that P explains H, but $P \cup \{X\}$ does not explain H.

Proof. We prove this property with a concrete example that we have encountered on one of our real data sets. Let A, B, C and H be four subspace clusters so that $AS(A) = 49$, $vol(A) = 0.0148305$; $AS(B) = 33$, $vol(B) = 0.0047081$; $AS(C) = 40$, $vol(C) = 0.00297933$; $AS(H) = 40$, $vol(H) = 0.0642656$. Furthermore, $vol(B \cap H) = 0$, $vol(A \cap C) = 0$, $vol(B \cap C) = 0$, $vol(C \cap H) = 0$. The total number of points in the data set is $n = 150$.

We show that $\{A, B\}$ explains H. We solve system (5.21) for A, B, and the uniform background noise, and we obtain the solution (n_A, n_B, n_U), where n_A and n_B represent the number of points generated by the component densities associated with A and B, respectively, and n_U is the number of points generated by the component density associated with the uniform background noise. We obtain $n_A = 41$, $n_B = 29$, and $n_U = 80$. At significance level $\alpha_0 = 1.0E - 10$, the left and right critical values of the Binomial distributions in (5.20) are: 0 and 18 for $Binomial(41, \frac{vol(A \cap H)}{vol(A)})$; 0 and 1 for $Binomial(29, \frac{vol(B \cap H)}{vol(B)} = 0)$; and 0 and 24 for $Binomial(80, vol(H))$. By summing up these bounds, we obtain the range $[0, 43]$. Because $AS(H) = 40 \in [0, 43]$, it follows that $\{A, B\}$ explains H.

We show that $\{A, B, C\}$ does not explain H. We solve system (5.21) for A, B, C, and the uniform background noise, and we obtain the solution (n_A, n_B, n_C, n_U), where n_A, n_B and n_C represent the number of points generated by the component densities associated with A, B, and C, respectively, and n_U is the number of points generated by the component density associated with the uniform background noise. We obtain $n_A = 41$, $n_B = 29$, $n_C = 39$, and $n_U = 41$. At significance level $\alpha_0 = 1.0E - 10$, the left and right critical values of the Binomial distributions in (5.20) are: 0 and 18 for $Binomial(41, \frac{vol(A \cap H)}{vol(A)})$; 0 and 1 for $Binomial(29, \frac{vol(B \cap H)}{vol(B)} = 0)$; 0 and 1 for $Binomial(39, \frac{vol(C \cap H)}{vol(C)} = 0)$; and 0 and 17 for $Binomial(41, vol(H))$. By summing up these bounds, we obtain the range $[0, 37]$. Because $AS(H) = 40 > 37$, it follows that $\{A, B, C\}$ does not explain H.

5.6 Redundancy-aware Problem Definition

The problem of representing R via a smallest (in this sense non-redundant) set of subspace clusters can now be defined.

Redundancy-aware problem definition. Given a data set D of n d-dimensional points. Let R be the set of all subspace clusters of D. Find a non-empty set $P^{opt} \subseteq R$ with smallest cardinality $|P^{opt}|$ so that P^{opt} explains H for all $H \in R$.

Property 5.7 The optimization problem is guaranteed to have a solution.

Proof. Based on Property 5.3, R explains H, $\forall H \in R$. Thus, if there is no other $P \subset R$ so that P explains H, $\forall H \in R$, the optimization problem has the solution R.

We note that, in general, the solution to the redundancy-aware problem definition may not be unique.

We note that any proper subset P' of P^{opt} does not have the property that it explains all subspace clusters in R. If such a proper subset P' would exist, then P' will be the optimal solution, because P' has less elements that P^{opt}, which contradicts the fact that P^{opt} is the optimal solution. In this sense, P^{opt} is non-redundant.

We emphasize the fact that the redundancy-aware problem definition avoids shortcomings of existing problem definitions in the literature. First, our objective is formulated through an optimization problem, which is independent of a particular algorithm used to solve it. Second, our definition of subspace cluster is based on statistical principles; thus, we can trust that P^{opt} stands out in the data in a statistical way, and is not simply an artefact of the method.

Enumerating all elements in R in an exhaustive way is computationally infeasible for larger values of n and d. Finding a smallest set of explaining subspace clusters by testing all possible subsets of R has complexity $2^{|R|}$, which is in turn computationally infeasible for typical sizes of R.

As a sanity check, we ran an exhaustive search on several small data sets where some low dimensional subspace clusters were embedded into higher dimensional spaces, similar to and including the data set depicted in Figure 5.1(g). The result sets P^{opt} found for these data sets were always containing only the embedded subspace clusters (i.e., we did not even have any false positives in these cases); In Figure 5.1(g), the two depicted 2-dimensional rectangles indicating the embedded subspace clusters represent in fact the subspace clusters found by the exhaustive search.

The redundancy-aware problem definition is an optimization problem. We consider the corresponding decision problem, called *RedundancyAware-k*:

RedundancyAware-k: Given a data set D of n d-dimensional points. Let R be the set of all subspace clusters of D. Given k an integer, $k \in \{1, \ldots, |R|\}$. Determine if there exists a set $P \subseteq R$ with $|P| = k$ so that P explains H for all $H \in R$.

We note that, given a set $P \subseteq R$, we can verify in polynomial-time, i.e., $O(|R|)$, if P is a solution of *RedundancyAware-k*. In addition, a naive algorithm for solving the decision problem *RedundancyAware-k* would generate all subsets P of R of length k, and check for each one of them whether P explains H, $\forall H \in R$. The running time of this algorithm is $O(|R| * \mathit{choose}(|R|, k))$, which is super-polynomial when k is near $|R|/2$, because the maximum value of $\mathit{choose}(|R|, k))$ is achieved when $k = \lfloor |R|/2 \rfloor$ or $k = \lceil |R|/2 \rceil$.

Based on these facts, we conjecture the NP-completeness of the decision problem *RedundancyAware-k*.

Relationship to data generation model. In the following, we assume that in a data set D with n d-dimensional data points, we embed K subspace clusters $\{C_1, \ldots, C_K\}$ according to Definition 5.3. The embedded subspace clusters may share relevant attributes and they may overlap in common relevant attributes. However, if two or more embedded subspace clusters have the same set of relevant attributes, then, they can overlap in some relevant attributes, but not in all, because in this case, the overlapping subspace clusters can and should be regarded as just one subspace cluster. In addition, we assume that the remaining data points that are not cluster points are uniformly distributed on each attribute of D.

Let $P^{model} \subseteq R$, $P^{model} = \bigcup_{i=1}^{K} C_i \cup \epsilon$, where ϵ are the false positives subspace clusters that may result from the statistical tests. Ideally, the optimal solution P^{opt} of the redundancy-aware problem definition should be P^{model}. This is because the existence of the subspace clusters in $R \backslash P^{model}$ is due to the existence of the subspace clusters in P^{model}, and, thus, P^{model} should explain all subspace clusters in R. In addition, the existence of a subspace cluster in P^{model} is not due to the existence of another subspace cluster in P^{model}, and thus, the subspace clusters in P^{model} should not explain each other. This indicates that P^{model} should be the smallest set of subspace clusters in R that explains all subspace clusters in R.

In general, we cannot guarantee that P^{opt} coincides with P^{model}. In fact, P^{opt} is likely to be "close" to P^{model} in the sense that the subspace clusters in P^{opt} correspond well to the subspace clusters in P^{model}, but they may have a few more or less data points in comparison with the subspace clusters in P^{model}. Figure 5.4 illustrates this point.

Figure 5.4 illustrates an embedded subspace cluster A in a subspace S, and two subspace clusters B and C in the same subspace. B and C consist of some of the points in A and some points from the uniform background surrounding

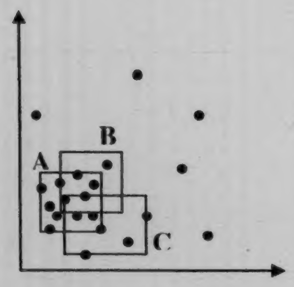

Figure 5.4: The existence of subspace clusters B and C is due to the existence of embedded subspace cluster A

A in subspace S. The existence of the subspace clusters B and C is due to the existence of the embedded subspace cluster A. In addition to B and C, there may be other subspace clusters in subspace S whose existence is due to the existence of the embedded subspace cluster A. However, we illustrate only B and C for better clarity in the figure.

A explains the subspace clusters that exist because A exists, such as B and C, and some of these subspace clusters may explain A as well. But the optimal solution P^{opt} is the *smallest* set with the property that it explains all subspace clusters in R. Thus, from the group of A and the subspace clusters whose existence is due to the existence of A, only one subspace cluster will be placed in P^{opt}. This subspace cluster may not be necessarily A; it may be another subspace cluster that differ from A in a few data points.

Chapter 6

Approximation Algorithm STATPC

In order to find the solution P^{opt} to the redundancy-aware problem definition, we need heuristics to 1) find a good set $R^{reduced} \subset R$ in which we can efficiently search for 2) a smallest set P^{sol} that explains (at least) all elements in $R^{reduced}$. Ideally, $P^{opt} \subseteq R^{reduced}$. We propose an approximation algorithm, called STATPC, that follows this schema.

Let $P^{opt} = \{T_1, \ldots, T_K\}$ be a solution to the redundancy-aware problem definition. We refer to the subspace clusters in P^{opt} as the "true" subspace clusters. The approximation algorithm STATPC first constructs a set $R^{reduced}$ by trying to find true subspace clusters around data points. Second, we solve heuristically the optimization problem on $R^{reduced}$ through a greedy optimization strategy and we obtain the solution P^{sol}.

The pseudo-code of STATPC is given in Figure 6.1.

6.1 Finding True Subspace Clusters Around Data Points

For a given data point Q, we want to determine if a true subspace cluster exists around Q. Towards this goal, first we select several *candidate* subspaces around Q, so that, when a true subspace cluster exists around Q, the probability that at least one of the candidate subspaces is relevant for the true subspace cluster is high. Second, for each candidate subspace, we find a locally optimal subspace cluster around Q. Third, since several locally optimal subspace clusters around Q in candidate subspaces may have been detected, we select the "best" one among them as the locally optimal subspace cluster among them.

```
Input: Data set $D = \{(x_{i1}, \ldots, x_{id}) | 1 \leq i \leq n\}$, parameters $\alpha_0$, $\alpha_K$, $\alpha_H$.
Output: Several, possibly overlapping, subspace clusters, and outliers.
Method:
    1. Build a set $R^{reduced}$:
        (a) Select a data point $Q$ from the set of data points that does not belong to subspace clusters detected so far, either randomly or based on a previous recommendation, if it exists.
        (b) Detect several candidate subspaces around $Q$ (Figure 6.3).
        (c) For each candidate subspace, detect a locally optimal subspace cluster around $Q$ (Figure 6.4).
        (d) Among the locally optimal subspace clusters detected in steps 1.b) and 1.c), one locally optimal subspace cluster is selected, and stored in $R^{reduced}$.
        (e) Repeat steps 1.a), 1.b), 1.c), and 1.d) until no data point can be selected for further subspace cluster search.
    2. Solve greedily the redundancy-aware problem on $R^{reduced}$, and obtain a solution $P^{sol}$ (Figure 6.5). Points that do not belong to any of the subspace clusters in $P^{sol}$ are declared outliers.
```

Figure 6.1: Pseudo-code of STATPC

6.1.1 Detecting Candidate Subspaces

To determine a candidate subspace around ~~a~~ given data point Q, we consider all $2D$ hyper-rectangles with Q in the center and side width $2 * \delta$, i.e., all hyper-rectangles:

$$[Q.Attr_i - \delta, Q.Attr_i + \delta] \times [Q.Attr_j - \delta, Q.Attr_j + \delta] \tag{6.1}$$

for some $\delta \in [0, 0.5]$, $1 \leq i < j \leq d$.

Subsequently, we propose to rank the $2D$ subspaces of the data set D in decreasing order of the actual support of these $2D$ hyper-rectangles. Let *Rank* denote this ranking. By analyzing the ranking *Rank*, we want to determine a set of attributes, called *signaled* attributes, which are, with high probability, relevant for one of the true subspace clusters around Q.

Motivation Let Q be a data point centrally located in at least one true subspace cluster. Then, $2D$ subspaces that involve attributes of the true subspace cluster(s) around Q are likely to be placed towards the top of the ranking *Rank*.

First, we consider the case when there is a single true subspace cluster C_1 in the data, which consists of n_1 data points, and has r_1 relevant attributes.

We estimate the actual support of a $2D$ hyper-rectangle H around Q that involves at least one relevant attribute of C_1. There are $choose(r_1, 2) + r_1 * (d - r_1)$ such $2D$ hyper-rectangle H, where d is the dimensionality of the data set.

Since Q belongs to C_1, H includes a certain fraction f_1 of the cluster points n_1, $f_1 \in (0, 1]$. The value of f_1 depends on the positioning of Q within C_1, and on the parameter δ. Ideally, $f_1 = 1$, when H includes all n_1 cluster points. Since the remaining $n - n_1$ data points are uniformly distributed on $subspace(H)$, H includes data points from the remaining data points proportionally to its volume $vol(H) = (2 * \delta)^2$. Thus, the actual support of H is estimated as:

$$AS(H) \approx f_1 * n_1 + (n - n_1) * (2 * \delta)^2 = (f_1 - (2 * \delta)^2) * n_1 + n * (2 * \delta)^2 \tag{6.2}$$

We estimate the actual support of a $2D$ hyper-rectangle H' around Q that does *not* involve any of the relevant attributes of C_1. There are $choose(d, 2) - [choose(r_1, 2) + r_1 * (d - r_1)]$ such $2D$ hyper-rectangle H'.

In this case, all n data points are uniformly distributed in $subspace(H')$, and the actual support of H' is estimated as:

$$AS(H') \approx n * (2 * \delta)^2 \tag{6.3}$$

Based on Equations (6.2) and (6.3), it follows that $AS(H) > AS(H')$ when $f_1 > (2 * \delta)^2$. For instance, for $\delta \in \{0.05, 0.1, 0.15\}$, $AS(H) > AS(H')$ when H includes at least a fraction $f_1 \in \{0.01, 0.04, 0.09\}$, respectively, of the cluster points n_1.

We cannot guarantee that condition $f_1 > (2 * \delta)^2$ always holds, because it may hold or not depending on the positioning of Q within C_1 and the value of δ. But, the more centrally located Q is within C_1, the more likely that condition $f_1 > (2 * \delta)^2$ holds.

Thus, given a data point Q centrally located in C_1, the actual support of a $2D$ hyper-rectangle around Q that involves at least one relevant attribute for C_1 is likely to be higher than the actual support of a $2D$ hyper-rectangle around Q that involves *no* relevant attribute for C_1. This is in fact equivalent to the statement of Property 6.1.

Second, we consider the case when there are K true subspace clusters in the data $C_1, C_2, \ldots, C_K$, so that each one of them consists of n_k data points, and has r_k relevant attributes, $1 \leq k \leq K$. Without restricting the generality, we can assume that Q belongs C_1.

As in the first case, we estimate the actual support of a $2D$ hyper-rectangle H around Q that involves at least one relevant attribute of C_1.

H includes a certain fraction f_1, $f_1 \in (0, 1]$, of the cluster points n_1. Some other true subspace clusters k, $2 \leq k \leq K$, may have relevant attributes in *subspace*(H); let B be the set of such subspace clusters. Let f_k, $f_k \in [0, 1]$, denote the fraction of points from true subspace cluster k in B that is included in H, $2 \leq k \leq K$.

The remaining $n - n_1 - \sum_{k \in B} n_k$ data points are uniformly distributed in *subspace*(H). The actual support of H is estimated as:

$$AS(H) \approx f_1 * n_1 + \sum_{k \in B} f_k * n_k + (n - n_1 - \sum_{k \in B} n_k) * (2 * \delta)^2$$

$$AS(H) \approx (f_1 - (2 * \delta)^2) * n_1 + \sum_{k \in B} (f_k - (2 * \delta)^2) * n_k + n * (2 * \delta)^2 \qquad (6.4)$$

We also estimate the actual support of a $2D$ hyper-rectangle H' around Q that does *not* involve any of the relevant attributes of C_1.

Some other true subspace clusters k, $2 \leq k \leq K$, may have relevant attributes in *subspace*(H'); let B' be the set of such subspace clusters. Let e_k, $e_k \in [0, 1]$, denote the fraction of points from true subspace cluster k in B' that is included in H', $2 \leq k \leq K$.

The remaining $n - \sum_{k \in B'} n_k$ data points are uniformly distributed in *subspace*(H'). The actual support of H' is estimated as:

$$AS(H') \approx \sum_{k \in B'} e_k * n_k + (n - \sum_{k \in B'} n_k) * (2 * \delta)^2$$

$$AS(H') \approx \sum_{k \in B'} (e_k - (2 * \delta)^2) * n_k + n * (2 * \delta)^2 \qquad (6.5)$$

In Equations (6.4) and (6.5), the amounts $\sum_{k \in B} (f_k - (2 * \delta)^2) * n_k$ and $\sum_{k \in B'} (e_k - (2 * \delta)^2) * n_k$, are likely comparable; and thus, $AS(H) > AS(H')$

likely holds when $f_1 > (2 * \delta)^2$. From here, we can draw the conclusion as in the first case.

We note that the above analysis suggests that Property 6.1 is the more likely to happen 1) the more points C_1 has, because, the larger n_1, the higher the chance that $AS(H) > AS(H')$; and 3) the more relevant attributes C_1 has, because the larger r_1, the more $2D$ hyper-rectangles that involve at least one relevant attribute of C_1.

There are $choose(d, 2) = \frac{d*(d-1)}{2}$ pairs of attributes in $Rank$, and let M be a positive integer, $1 \leq M \leq \frac{d*(d-1)}{2}$. Property 6.1 does not mean that the top M pairs in $Rank$ consist only of relevant attributes for the true subspace cluster(s) around Q. Property 6.1 means that the *frequency* with which relevant attributes for the true subspace cluster(s) around Q occur in the top M pairs in $Rank$ is likely to be significantly higher than the frequency with which irrelevant attributes for the true subspace cluster(s) around Q occur in top M pairs in $Rank$. Therefore, if we measure the frequency with which individual attributes occur in the top M pairs in $Rank$, then, attributes with "high" frequency are highly likely to be relevant attributes for the true subspace cluster(s) around Q. We need a way to decide which frequencies are "higher than expected".

To determine if an attribute $a \in A$ is significantly more frequent than expected in the top M pairs in $Rank$, we compare its frequency to the expected frequency of an attribute in a *randomly* selected set of M pairs from the set of all pairs of attributes formed with the attributes in A.

In a data set where all data points are uniformly distributed on all attributes, the actual support of a $2D$ hyper-rectangle of side width $2 * \delta$ around a data point Q is given by formula (6.3). Thus, if we rank the $2D$ subspaces of the data set in decreasing order of the actual support of these $2D$ hyper-rectangles, any ranking is equally likely. Then, taking the top M pairs in such a ranking is equivalent to selecting M pairs of attributes *randomly* from the set of all pairs of attributes formed with the attributes in A.

Let $a \in A$ be an attribute of the data set D. Let us assume that we select M pairs of attributes *randomly* from the set of all pairs of attributes formed with the attributes in A. Let X be the random variable that represents the number of occurrences of attribute a in the selected M pairs of attributes. X is a hyper-geometric distributed variable with parameters $\frac{d*(d-1)}{2}$ (number of all pairs), $d-1$ (number of pairs containing attribute a), and M (sample size): for $1 \leq k \leq M$, $Pr(X = k)$ equals

$$Pr(X = k) = \frac{choose(d-1, k) * choose(\frac{d*(d-1)}{2} - (d-1), M-k)}{choose(\frac{d*(d-1)}{2}, M)} \qquad (6.6)$$

Definition 6.1 Given a data set D of n d-dimensional points. Let M be a positive integer, $1 \leq M \leq \frac{d*(d-1)}{2}$. Let α_H be a significance level. Let θ_{α_H} be the right critical value of the hyper-geometric distribution (6.6), at significance level α_H. An attribute $a \in A$ is said to *occur more often than expected* in the top M pairs in *Rank*, if its number of occurrences in the top M pairs in *Rank* is larger than θ_{α_H}.

We have to decide a value for M. M should take relatively small values, because, as we go down the ranking, eventually all attributes will appear as often as expected.

We observe that, given a fixed significance level value α_H, different values of M result in the same right critical values θ_{α_H} for the hyper-geometric distribution (6.6), because of the discrete nature of the distribution. For instance, if $\alpha_H = 0.001$, $d = 50$, then $\theta_{\alpha_H} = 2$ for $M \in M^{val} = \{2, 3, 4, 5\}$. This means that, for any value of M in the set M^{val}, we will conclude that an attribute $a \in A$ occurs more often than expected in the top M pairs in *Rank*, if it occurs at least $\theta_{\alpha_H} = 2$ times in the top M pairs in *Rank*. Therefore, we shall choose M as the largest value in M^{val}.

In STATPC, in order to be robust to the value of M, and in order to position M at the top of the ranking, we consider three sets of values M_1^{val}, M_2^{val}, and M_3^{val} for M that result in three consecutive critical values $\theta_{\alpha_H} \in \{2, 3, 4\}$. In each case, we set M to the largest value in M_i^{val}, $1 \leq i \leq 3$, and we obtain three values M_i, $1 \leq i \leq 3$, for M. In our example, the three values for M are 5, 12, and 20, because for each $M \in \{2, 3, 4, 5\}$, we obtain $\theta_{\alpha_H} = 2$; for each $M \in \{6, 7, 8, 9, 10, 11, 12\}$, we obtain $\theta_{\alpha_H} = 3$; and for each $M \in \{13, 14, 15, 16, 17, 18, 19, 20\}$, we obtain $\theta_{\alpha_H} = 4$.

Definition 6.2 For each M_i, $1 \leq i \leq 3$, let A_i, $1 \leq i \leq 3$, be sets of attributes that occur more often than expected in the top M_i, $1 \leq i \leq 3$ pairs in *Rank*. For an attribute $a \in A$, we define *count*(a) as the number of times a occurs in *all* A_i, $1 \leq i \leq 3$. We define the *signaled* attributes as $SignaledAttributes = \{a \in A | count(a) > 0, count(a) = max_{b \in A} count(b)\}$.

For example, if we obtain $A_1 = \{Attr_1, Attr_2, Attr_3\}$ for $M_1 = 5$; $A_2 = \{Attr_1, Attr_2, Attr_3\}$ for $M_2 = 12$; and $A_3 = \{Attr_1, Attr_3\}$ for $M_3 = 20$, then the set of signaled attributes is $\{Attr_1, Attr_3\}$.

By taking the signaled attributes to be the most frequently occurring attributes in A_i, $1 \leq i \leq 3$, we decrease the probability that a signaled attribute is irrelevant for all true subspace clusters to which Q belongs, and we increase the probability that a signaled attribute is relevant for the true subspace cluster(s) to which Q belongs.

We note that, if an attribute $a \in A$ occurs more often than expected in the top M_1 pairs in *Rank*, it is not guaranteed that attribute a occurs more often than expected in the top M_2 pairs in *Rank*, where $M_2 > M_1$. In our previous example, it is possible that when we take the top $M_1 = 5$ pairs in *Rank*, attribute a occurs more often than expected because it occurs in 2 pairs, but when we take the top $M_2 = 12$ pairs in *Rank*, attribute a does not occur more

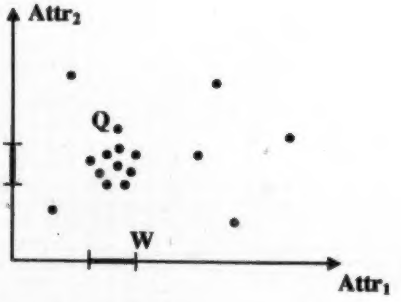

Figure 6.2: The issue of committing to a candidate subspace

often than expected, because it occurs in the same 2 pairs, and in order to occur more often than expected, it should have occurred in at least 3 pairs.

Iterative refinement of signaled attributes. Let S^0 be a set of signaled attributes. We observe that if S^0 is only a subset of the relevant attributes for a true subspace cluster around Q, then, by considering the points in a hyper-rectangle W of side width $2 * \delta$ around Q in subspace S^0, we capture a fraction of the true subspace cluster's points, which is often large enough to allow us to determine more of the relevant attributes; these are attributes where the points in *SuppSet*(W) are *not* uniformly distributed.

Based on this observation, we can obtain a *candidate subspace* around Q through an iterative refinement of S^0, as follows. Let S^1 be the set of relevant attributes for W built in subspace S^0. If S^0 is *not* included in S^1, return the empty set. If $S^0 = S^1$, return S^0. Otherwise, we repeat with S^1, selecting the relevant attributes of W built in subspace S^1, and so on, until no more attributes can be added.

Commit to a candidate subspace or recommend the next data point. Let $S = S^{iter}$, $iter \geq 1$, $S \neq \emptyset$, be the candidate subspace determined by the iterative refinement of a set of signaled attributes S^0. Let W be a hyper-rectangle of side width $2 * \delta$ around Q in subspace S^{iter-1}.

By the construction of S, the data points in *SuppSet*(W) are *not* uniformly distributed on each attribute $a \in S$. Thus, for each attribute $a \in S$, we would like to detect the 1D regions that are responsible for the fact that the data points in *SuppSet*(W) are not uniformly distributed on a. In addition, Q may or may not belong to these 1D regions.

For instance, in Figure 6.2, Q is a data point for which we want to determine a candidate subspace. We determine that the set of signaled attributes S^0 is $S^0 = \{Attr_1\}$. W is a hyper-rectangle of side width $2 * \delta$ around Q in subspace $S^0 = \{Attr_1\}$. Through the iterative refinement, S^0 is extended into the candidate subspace $S = \{Attr_1, Atttr_2\}$, because the points in *SuppSet*(W) are not uniformly distributed on $Attr_1$ and on $Attr_2$. The 1D regions that are responsible for the fact that the points in $SuppSet(W)$ are not uniformly distributed on $Attr_1$ and on $Attr_2$ are depicted as bold segments on $Attr_1$ and on $Attr_2$. Q belongs to such a 1D region on the signaled attribute $Attr_1$, but it does not belong to such a 1D region on the attribute $Attr_2$.

As exemplified in Figure 6.2, there are cases when a candidate subspace contains a subspace cluster (which may be a true subspace cluster), and Q is placed in the candidate subspace in the vicinity of the subspace cluster. If we keep the candidate subspace, then, in the next step, STATPC will compute a locally optimal subspace cluster around Q in the candidate subspace. Because of the positioning of Q with respect to the subspace cluster, the locally optimal subspace cluster around Q in the candidate subspace will be a subspace cluster that includes Q and some of the cluster points. The locally optimal subspace cluster around Q in the candidate subspace may be stored in $R^{reduced}$, in which case, its points cannot be selected for further subspace cluster search. Thus, the probability of selecting a data point for subspace cluster search that is centrally located in the subspace cluster of the candidate subspace has decreased. Therefore, the probability that the subspace cluster of the candidate subspace will be stored in $R^{reduced}$ has decreased as well.

Thus, if a situation like the one illustrated in Figure 6.2 is detected, we *do not commit* to the candidate subspace, i.e., we do not keep this subspace as a candidate subspace. Furthermore, we are able to *recommend* the next data point for subspace cluster search as the data point that does not belong to any of the previously computed subspace clusters, and that is located centrally in the subspace cluster of the candidate subspace. In this way, we increase the probability of having in $R^{reduced}$ the subspace cluster of the candidate subspace for the next data point used for subspace cluster search.

Therefore, we need to decide whether to commit or not to a candidate subspace S. For each attribute $a \in S$, we detect the 1D region(s) that are responsible for the fact that the data points in $SuppSet(W)$ are not uniformly distributed on a using a methodology similar to the one used in P3C to detect cluster projections. First, we divide attribute a into $\lfloor 1+log_2(AS(W)) \rfloor$ bins, by the Sturge's rule [65]. Second, we compute, for each bin, how many points from $SuppSet(W)$ it contains, and compare this number with the expected number of points in a bin if the points in $SuppSet(W)$ were uniformly distributed across all the bins on attribute a. If a bin has more points than expected, then the bin is marked. Finally, adjacent marked bins are merged into 1D regions. For an attribute $a \in S$, more than one such $1D$ region may be detected on a.

If there exists at least one attribute $a \in S$ so that Q does *not* belong to one 1D region, computed as above, on this attribute, then, we conclude that, if there were a subspace cluster in the candidate subspace, then Q is placed in its vicinity, and thus we *do not commit* to the candidate subspace.

When we do not commit to a candidate subspace, we can recommend the next data point for subspace cluster search. The 1D regions identified as described above form one or more multi-dimensional regions in the candidate subspace. We choose arbitrarily one of the multi-dimensional regions in the candidate subspace, and we recommend as the next data point for subspace cluster search, the data point that does not belong to any of the previously computed subspace clusters, and that is closest, in terms of Manhattan distances in the candidate subspace, to the centroid of the multi-dimensional region formed with the 1D regions.

So far, we have detected a candidate subspace around Q given a certain value for δ. There is no "best" value for δ, and to improve our chances of detecting a true subspace cluster if it exists around a data point, we suggest to use several different values. We simply try the 3 values 0.05, 0.1, 0.15 for δ. The candidate subspaces detected for different values of δ may be identical or they may be the empty set $\emptyset$; thus, we detect *up to* 3 candidate subspaces for each data point Q that we consider.

6.1.2 Detecting a Locally Optimal Subspace Cluster per Candidate Subspace

Let S be a candidate subspace. To determine if a subspace cluster around Q exists in S, we build a series of MBRs in S, starting from Q, and adding in each step to the current MBR the data point with the smallest MINDIST to the current MBR in subspace S. MINDIST [1] is the popular distance between a data point and an MBR used in index structures [39]. For efficiency reasons, and because a cluster contains typically only a fraction of the total number of points, we only build $0.3 * n$ MBRs around Q in subspace S.

Let R^{local} be the set of MBRs built in this way that are also subspace clusters. If $R^{local} = \emptyset$, no true subspace cluster around Q in S could be found; otherwise, we obtain a set of highly overlapping subspace clusters. We want to select *one* of these subspace clusters that is *locally optimal* in the sense that it explains *more* subspace clusters in R^{local} than any other subspace cluster in R^{local}. However, because the subspace clusters in R^{local} are highly redundant, there may be several subspace clusters in R^{local} that explain the same maximum number of subspace clusters in R^{local}. Thus, we want to select *one* of these subspace clusters that is *locally optimal* in the sense that it explains *better* all subspace clusters in R^{local} than any other subspace cluster in R^{local}.

Under the same data model assumptions as for the *Explain* relationship, we can define the *expected support* of a subspace cluster H assuming a single

---

[1] If $l = (l_1, \ldots, l_d)$ and $u = (u_1, \ldots, u_d)$ are the left-most, respectively, right-most corners of a d-dimensional MBR M, then MINDIST between a d-dimensional point $P = (p_1, \ldots, p_d)$ and the MBR M is the square of $\sum_{i=1}^{d}(p_i - r_i)^2$, where $r_i = l_i$, if $p_i < l_i$; $r_i = u_i$, if $p_i > u_i$; $r_i = p_i$, else.

```
Input: Data set $D = \{(x_{i1}, \ldots, x_{id}) | 1 \leq i \leq n\}$, parameter $\alpha_H$, data point $Q$.
Output: Up to 3 candidate subspaces around $Q$.
Method:
    1. For each $\delta \in \{0.05, 0.1, 0.15\}$:
        (a) Build a hyper-rectangle of side width $2 * \delta$ around $Q$ in each $2D$ subspace of the data set.
        (b) Rank the 2D subspaces in decreasing order of the actual support of the 2D hyper-rectangles built at step 1.a).
        (c) Let $\theta_{\alpha_H}$ be the right critical value of the hyper-geometric distribution (6.6) at significance level $\alpha_H$. Determine the largest values $M_1, M_2, M_3$ for which the corresponding critical values $\theta_{\alpha_H}$ are 2, 3 and 4, respectively.
        (d) For each $M_i$, $1 \leq i \leq 3$, determine a set of attributes $A_i$, $1 \leq i \leq 3$, that occur more often than expected in the top $M_i$ pairs in the ranking computed in step 1.b), i.e., attributes that occur more than 2, 3, and 4, respectively, times in the top $M_i$ pairs in the ranking computed in step 1.b).
        (e) Compute the set of signaled attributes $S^0$ as the attributes most frequent in $A_i$, $1 \leq i \leq 3$.
        (f) Refine iteratively $S^0$ and obtain a candidate subspace $S$:
            $S \leftarrow \emptyset$; $iter \leftarrow 0$
            while($Continue$)
            $W :=$ hyper-rectangle of side width $2 * \delta$ around $Q$ in $S^{iter}$
            $S^{iter+1} :=$ relevant attributes of $W$ built in $S^{iter}$
            If $S^{iter}$ not included in $S^{iter+1}$: $Continue = 0$, return $S$
            If $S^{iter} = S^{iter+1}$: $Continue = 0$, return $S = S^{iter}$
            If $S^{iter} \subset S^{iter+1}$: $S \leftarrow S^{iter+1}$, $iter \leftarrow iter + 1$
            End while
        (g) Decide whether to commit to candidate subspace $S$, and if not, recommend the next data point for candidate subspace construction.
```

Figure 6.3: Pseudo-code of detecting candidate subspaces around a point Q

subspace cluster P_1 and the uniform background noise, as:

$$ES(H|P_1) = n_{P_1} * \frac{vol(H \cap P_1)}{vol(P_1)} + (n - n_{P_1}) * vol(H) \tag{6.7}$$

where n_{P_1} is the number of points generated by the density component associated with P_1, obtained by solving Equation (5.21) for P_1 and background noise.

We measure how well P_1 explains H by comparing expected support $ES(H|P_1)$ and actual ~~support~~ of $AS(H)$, using a function $QualityExplain : R^{local} \times R^{local} \rightarrow [0,1]$:

$$QualityExplain(P_1, H) := 1 - \frac{|AS(H) - ES(H|P_1)|}{max(AS(H), ES(H|P_1))}, \quad P_1, H \in R^{local} \tag{6.8}$$

$QualityExplain$ represents the relative difference between the actual support $AS(H)$ of H and the estimated support $ES(H|P_1)$ of H given the subspace cluster P_1 and the uniform background noise. $QualityExplain(P_1, H)$ can be written as:

$$QualityExplain(P_1, H) = \begin{cases} \frac{AS(H)}{ES(H|P_1)}, & if\ AS(H) < ES(H|P_1) \\ \frac{ES(H|P_1)}{AS(H)}, & if\ AS(H) \geq ES(H|P_1) \end{cases}$$

The closer $AS(H)$ and $ES(H|P_1)$ are, the closer $QualityExplain$ is to 1, and the better the quality of explanation.

Consequently, we choose as the locally optimal subspace cluster around Q in S the subspace cluster $P^{local} \in R^{local}$ that maximizes

$$\sum_{H \in R^{local}} QualityExplain(P^{local}, H) \tag{6.9}$$

6.1.3 Detecting a Locally Optimal Subspace Cluster Between Locally Optimal Subspace Clusters in Candidate Subspaces

For a given data point Q, let R^{all_local} be the set of all locally optimal subspace clusters around Q detected in Sections 6.1.1 and 6.1.2. Since we determine up to 3 candidate subspaces around Q, and in each one of these candidates subspaces, we determine up to 1 locally optimal subspace cluster around Q, it holds that $|R^{all\_local}| \leq 3$.

We select the subspace cluster $P^{all\_local} \in R^{all\_local}$ that explains better all subspace clusters in R^{all_local} than any other subspace cluster in R^{all_local}. Formally, we select the subspace cluster $P^{all\_local} \in R^{all\_local}$ that maximizes

$$\sum_{H \in R^{all\_local}} QualityExplain(P^{all\_local}, H) \tag{6.10}$$

```
Input: Data set $D = \{(x_{i1}, \ldots, x_{id}) | 1 \leq i \leq n\}$, parameter $\alpha_0$, $\alpha_K$, data point $Q$, candidate subspace $S$.
Output: A subspace cluster around $Q$ in subspace $S$.
Method:
    1. Build $0.3 * n$ MBRs around $Q$ in subspace $S$ by adding, one at a time, the data point with smallest MINDIST to the current MBR.
    2. Build $R^{local}$ as the set of MBRs constructed in step 1) that are subspace clusters.
    3. Choose as the locally optimal subspace cluster around $Q$ in $S$ the subspace cluster $P^{local} \in R^{local}$ that maximizes (6.9).
```

Figure 6.4: Pseudo-code of detecting a locally optimal subspace cluster around a data point Q in a candidate subspace S.

6.1.4 Constructing the Set $R^{reduced}$

To construct a set $R^{reduced}$, STATPC tries to find subspace clusters around data points as described in Sections 6.1.1, 6.1.2, and 6.1.3. The first point to consider is selected randomly from the set of all points. Subsequent points are selected randomly from the points that do not belong to detected subspace clusters in previous steps. Building $R^{reduced}$ terminates when no data point can be selected for further subspace cluster search.

6.2 Greedy Optimization

Although $|R^{reduced}| < |R|$, solving the optimization problem on $R^{reduced}$ by testing all its possible subsets is still computationally too expensive in general. Thus, we construct *greedily* a set P^{sol} that explains all subspace clusters in $R^{reduced}$, but may not be the smallest set with this property.

We build P^{sol} by adding one subspace cluster at a time from $R^{reduced}$. At each step, let *Cand* be the set of subspace clusters in $R^{reduced}$ that are not explained by the current P^{sol}. Thus, subspace clusters in *Cand* can be used to extend P^{sol} further, until P^{sol} explains all subspace clusters in $R^{reduced}$. Initially, $P^{sol} = \emptyset$ and $Cand = R^{reduced}$. In each step, we select the subspace cluster $H^* \in Cand$ for which $P^{sol} \cup \{H^*\}$ explains more subspace clusters in $R^{reduced}$ than any other choice from the set *Cand*. H^* is added to P^{sol}, and we stop when *Cand* is empty.

Because of Property 5.3, set *Cand* cannot include a subspace cluster that has been already selected in P^{sol}. Thus, *Cand* is guaranteed to become void, and the optimization strategy is guaranteed to end.

```
Input: Data set $D = \{(x_{i1}, \dots, x_{id}) | 1 \le i \le n\}$, parameter $\alpha_0$, set $R^{reduced}$.
Output: A solution $P^{sol} \subseteq R^{reduced}$ so that $P^{sol}$ explains $H$, $\forall H \in R^{reduced}$.
Method:
1. Initialization: $P^{sol} := \emptyset$; $Cand := R^{reduced}$.
2. Greedy optimization:
While ($Cand \neq \emptyset$)
    Choose $H^* \in Cand$ so that $P^{sol} \cup \{H^*\} = argmax_{H \in Cand} |ExplainedBy(P^{sol} \cup H)|$.
    $P^{sol} := P^{sol} \cup \{H^*\}$.
    $Cand := R^{reduced} \setminus ExplainedBy(P^{sol})$.
End while
```

Figure 6.5: Pseudo-code of detecting greedily a solution P^{sol} on $R^{reduced}$.

The pseudo-code of the greedy optimization is given in Figure 6.5. For $\forall P \subseteq R^{reduced}$, we define $ExplainedBy(P)$ as the set of subspace clusters in $R^{reduced}$ that are explained by P, i.e., $ExplainedBy(P) := \{H \in R^{reduced} | \ P$ explains $H\}$.

6.3 A Note on Parameters

The accuracy of STATPC depends mainly on 3 parameters: α_0, α_K, and α_H. All these parameters represent significance levels used in statistical tests.

STATPC takes three additional parameters. The first additional parameter is the number of values for M, where M is used to inspect the top of the ranking *Rank*. In our implementation, we consider 3 values for M. The actual values for M are well-determined based on the statistical significance level α_H. The second additional parameter is the value of δ for building $2D$ hyper-rectangles around a data point Q. We consider 3 values for δ in our implementation. The third additional parameter is the number of MBRs that we construct around a data point Q in a candidate subspace, which is currently set to 0.3. Some preliminary experiments suggested that the effect of varying these additional parameters is likely to be more pronounced in terms of computational time than in terms of accuracy of the algorithm.

The robustness of STATPC to the main parameters α_0, α_K, and α_H is studied in Section 6.5.8.

6.4 Theoretical Complexity

For a given data point Q, the complexity O_1 of building a candidate subspace around Q is controlled by: 1) the complexity of step 1.a) in Figure 6.3, which is $O(d * n + \frac{d*(d-1)}{2})$; 2) the complexity of step 1.b) in Figure 6.3, which is $O(\frac{d*(d-1)}{2} * log_2[\frac{d*(d-1)}{2}])$; 3) the complexity of step 1.f) in Figure 6.3, which is $O(d * no\_iter)$, where no_iter is the number of times we iteratively refine S^0; and 4) the complexity of step 1.g) in Figure 6.3, which is $O(dim(S))$.

For a given data point Q, the complexity O_2 of detecting a locally optimal subspace clusters around Q in candidate subspace S is controlled by: 1) the complexity of steps 1) and 2) in Figure 6.4, which is $O(0.3 * n * (n * dim(S) + dim(S))$; and 2) the complexity of step 3) in Figure 6.4, which is $O(|R^{local}|^2)$.

Thus, the complexity of building the set $R^{reduced}$ is $O(no\_data\_points\_tried * (3 * (O_1 + O_2) + |R^{all\_local}|^2))$.

The complexity of the greedy optimization is $O(|R^{reduced}| * 1^3 + \ldots + |R^{reduced}| * |P^{sol}|^3)$.

6.5 Experimental Evaluation

The experiments reported in this section were conducted on a Linux machine with 3 GHz CPU and 2 GB RAM.

6.5.1 Compared Techniques

As for P3C, we compare STATPC against several state-of-the-art projected and subspace clustering techniques: SSPC, PROCLUS, HARP, MINECLUS, MAFIA and ORCLUS. In addition to these techniques, we compare STATPC against P3C and PRIM.

We also compare STATPC against a representative set of full dimensional clustering algorithms: KMeans (denoted by KM), EM, CLARANS, agglomerative (BAHC) and divisive (DIANA) hierarchical clustering, and DBSCAN.

6.5.2 Synthetic Data

We study systematically the performance of the compared techniques as a function of different data generation criteria:

1. The distribution of cluster points in the relevant subspace: 1) uniform or 2) Gaussian;
2. The number of relevant attributes that clusters can have: 1) an equal or 2) a different number of relevant attributes;
3. The average number of relevant attributes;

4. The database dimensionality d;

5. Database size n;

6. Number of clusters k;

7. Cluster sizes and number of noise points;

8. Extent of clusters in their relevant attributes;

9. Overlap between clusters in common relevant attributes;

By combining the first 2 criteria, we obtain 4 categories of synthetic data sets: *Uniform_Equal*, *Uniform_Different*, *Gaussian_Equal*, and *Gaussian_Different*. For each category, we study the effect of the 3^{rd} criterion in data generation over the performance of the compared techniques. For this purpose, in each category, we generate data sets with $n = 300$ data points, $d = 50$ attributes, $k = 5$ clusters (clusters sizes are 60, 50, 40, 40, and 50 points), 60 uniformly distributed noise points, and the average number of relevant attributes in $\{2, 4, 6, 8, 10, 15, 20\}$. The clusters have axis-parallel orientation, i.e., when the cluster points are Gaussian distributed in their relevant subspace, the Gaussian distributions have diagonal covariance matrices, and when the cluster points are uniform distributed in their relevant subspace, the clusters are axis-parallel hyper-rectangles. Cluster points are uniformly distributed on $[0, 1]$ on the irrelevant attributes. The extent of clusters in their relevant attributes is between 10% and 30% of the attribute range. No overlap between clusters in common relevant attributes is introduced.

To study the effects of the remaining criteria in data generation, we generate synthetic data sets where the cluster points are *uniformly* distributed in their relevant subspace, and clusters have an *equal* number of relevant attributes (i.e., 4 relevant attributes per cluster), with the parameters listed above, and we vary the parameter of interest.

To study the effect of the database dimensionality (4^{th} criterion), for a database of size $n = 300$, $k = 5$ (60, 50, 40, 40, 50 cluster points, and 60 uniformly distributed noise points), 4 relevant attributes per cluster, we vary $d \in \{20, 35, 50, 75, 100\}$.

To study the effect of the database size in data generation (5^{th} criterion), we vary $n \in \{100, 300, 500, 1000, 2000\}$. Cluster sizes and number of noise points are as follows: for $n = 100$: 20, 17, 14, 14, 17 cluster points and 18 noise points; for $n = 300$: 60, 50, 40, 40, 50 cluster points and 60 noise points; for $n = 500$: 100, 84, 67, 67, 84 cluster points and 98 noise points; for $n = 1000$: 200, 170, 140, 140, 170 cluster points and 180 noise points; for $n = 2000$: 400, 340, 280, 280, 340 cluster points and 360 noise points.

To study the effect of the number of clusters in data generation (6^{th} criterion), we vary $k \in \{2, 3, 4, 5\}$. Cluster sizes and number of noise points are as follows: for $k = 2$: 125, 125 cluster points and 50 noise points; for $k = 3$:

83, 83, 84 cluster points and 50 noise points; for $k = 4$: 62, 63, 63, 62 cluster points and 50 noise points; for $k = 5$: 50, 50, 50, 50, 50 cluster points and 50 noise points.

To study the effect of the cluster sizes and number of noise points in data generation (7^{th} criterion), we vary cluster sizes and number of noise points as follows: Setup 1) 40, 30, 20, 20, 30 cluster points and 160 noise points; Setup 2) 50, 40, 30, 30, 40 cluster points and 110 noise points; Setup 3) 60, 50, 40, 40, 50 cluster points and 50 noise points; Setup 4) 65, 55, 45, 45, 55 cluster points and 35 noise points.

To study the effect of the extent of clusters in their relevant attributes in data generation (8^{th} criterion), we generate the clusters with Setup 1) 0.1, Setup 2) 0.2, Setup 3) 0.3, respectively Setup 4) 0.4 extent in the relevant attributes.

To study the effect of the overlap between clusters in common relevant attributes in data generation (9^{th} criterion), we generate data sets with $k = 2$ (125, 125 cluster points and 50 noise points), so that the two clusters are characterized by an overlap of Setup 1) 0, Setup 2) 0.1, Setup 3) 0.2, respectively Setup 4) 0.3 in common relevant attributes.

6.5.3 Real Data

We test the compared techniques on the following data sets from the UCI machine learning repository [67]: Pima Indians Diabetes (768 points, 8 attributes, 2 classes); Liver Disorders (345 points, 6 attributes, 2 classes); Wisconsin Breast Cancer Prognostic (WPBC)(198 points, 34 attributes, 2 classes); and Glass (214 points, 9 attributes, 6 classes).

6.5.4 Experimental Setup

STATPC has the same experimental setup as the one used in P3C.

In addition, for P3C, we use the variant that computes overlapping clusters. For KM and EM, we use the implementations available in the R statistical software [62]. BAHC, DIANA, and CLARANS are provided by the Biosphere project [78]. DBSCAN is provided by [3]. PRIM is available as a package [60] for the R statistical software.

For P3C, we set $\alpha\_Binom = 1.0E - 20$. For PRIM, we set $peel\_alpha = 0.05$, $paste\_alpha = 0.01$, $mass\_min = 0.1$. The full dimensional algorithms, except DBSCAN, require the target number of clusters as a parameter, which is set to the number of implanted clusters on synthetic data, and to the number of classes on real data. For CLARANS, we set $maxn = 250$, $numl = 5$. For DBSCAN, ϵ is set to 10% of the maximum distance in the data space, and *minpts* is set to 3.

STATPC requires 3 significance levels: α_0, α_K, and α_H. After testing the sensitivity of STATPC to these parameters (see Figures 6.21, 6.22, and 6.23), we set $\alpha_0 = 1.0E - 10$, $\alpha_K = \alpha_H = 0.001$.

As for P3C, the real data sets used are extended with attributes where the data points are uniformly distributed on $[0, 1]$. The real data sets do not contain missing values.

6.5.5 Performance Measures

We measure the accuracy of the compared techniques as in P3C using a F value, as defined in Section 3.8.5.

6.5.6 Statistical Significance of Results

STATPC computes subspace clusters that are statistically significant. The other techniques sometimes compute statistically significant subspace clusters, other times they do not, depending on parameter values and on the density of the implanted clusters (denser clusters are easier to detect). The classes in the real data sets form statistically significant clusters, and these clusters stay statistically significant when adding uniform attributes, as shown in Section 5.3.

6.5.7 Accuracy Results

Effect of average cluster dimensionality. Figure 6.6 shows the accuracy of the compared techniques as a function of increased average cluster dimensionality for the category $Uniform\_Equal$, where the cluster points are *uniformly* distributed in their relevant subspace, and the clusters have an *equal* number of relevant attributes. Figures 6.7, 6.8 and 6.9 illustrate the accuracy of the compared techniques as a function of increased average cluster dimensionality for the categories $Uniform\_Different$, $Gaussian\_Equal$, and $Gaussian\_Different$.

We observe that STATPC significantly and consistently outperforms the competing techniques, both in terms of clustering accuracy and in terms of accuracy of the found relevant attributes. The difference in accuracy between STATPC and previous techniques is more pronounced for the more difficult case of data sets with low dimensional subspace clusters.

We observe that the accuracies of SSPC, PROCLUS, HARP and MINECLUS increase as the average cluster dimensionality increases. PROCLUS depends strongly on an initial clustering in full dimensional space, which is a better approximation of the implanted clusters as the average cluster dimensionality increases, because the implanted clusters become more easily recognizable in full dimensional space. For the same reason, the accuracy of ORCLUS increases slightly as the average cluster dimensionality increases, but even when the average cluster dimensionality is 20, ORCLUS cannot estimate well enough the directions of least spread of the clusters. SSPC, HARP and MINECLUS

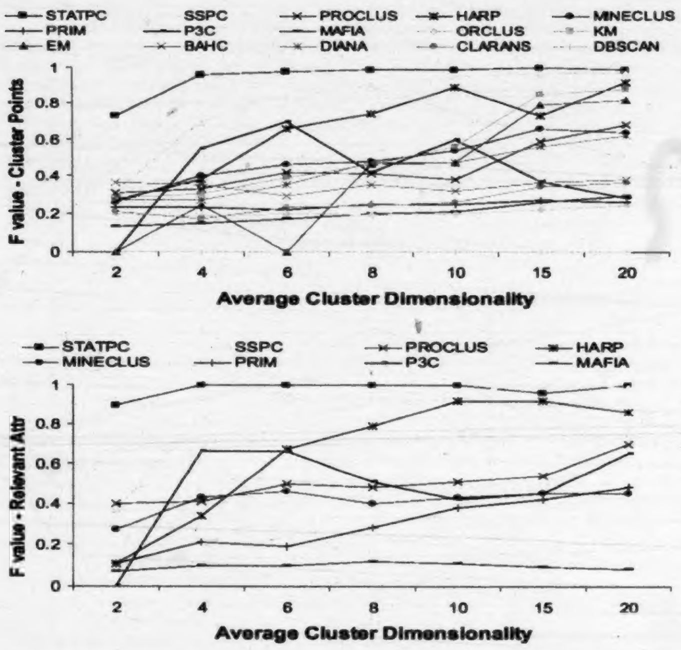

Figure 6.6: STATPC and competing techniques on category Uniform_Equal

leverage the principle that clusters with as many relevant attributes as possible are preferable; thus, their accuracies increase with increasing average cluster dimensionality.

The accuracies of MAFIA and PRIM increase only slightly as the average cluster dimensionality increases, but these accuracies have low values.

P3C does not exhibit increasing accuracy as the average cluster dimensionality increases. The reason is that, on these data sets, the density of the implanted clusters is low enough so that P3C cannot detect all the $1D$ projections of the implanted clusters, and the statistical evidence needed to aggregate $1D$ cluster projections is lacking.

The accuracies of the full dimensional clustering algorithms increase as the average cluster dimensionality increases. The reason is that the higher the average cluster dimensionality, the more recognizable are the implanted clusters in full dimensional space. EM may sometimes report an accuracy of 0, if it encounters in the computation singular or nearly-singular covariance matrices. We observe that the full dimensional clustering algorithms are not effective for the task of retrieving the implanted clusters, especially when the average cluster dimensionality is small. However, some of the full dimensional clustering algorithms outperform some of the projected and subspace clustering techniques, especially for higher average cluster dimensionality. DBSCAN was unable to detect any clusters although a reasonable parametrization was used.

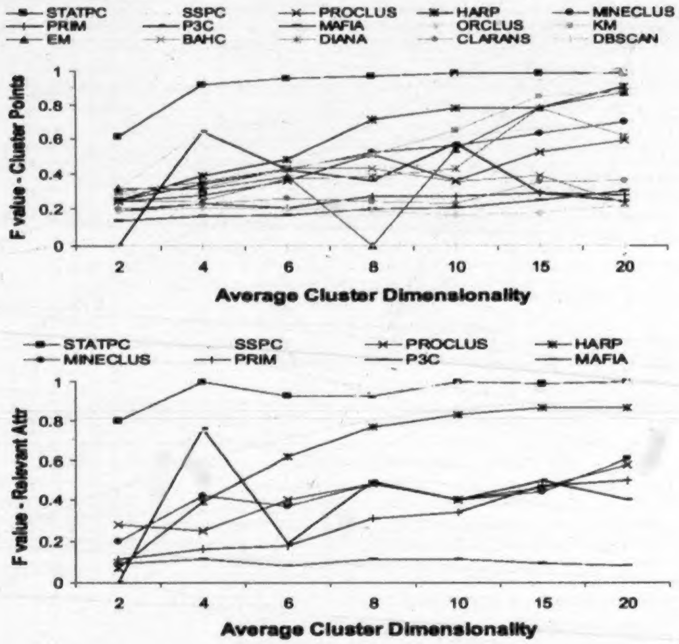

Figure 6.7: STATPC and competing techniques on category Uniform_Different

Note, however, that the remaining full dimensional algorithms are bound to find a predefined number of clusters.

Effect of distribution of cluster points in the relevant subspace. For all techniques, including STATPC, the accuracy results on data sets where cluster points are uniformly distributed in their relevant subspace are slightly higher than the accuracy results on data sets where cluster points are Gaussian distributed in their relevant subspace. The reason is that Gaussian distributed clusters are denser in the center than at the boundaries, and all techniques tend to recover only the dense, central part; thus, the decrease in accuracy.

Effect of equal vs. different number of relevant attributes. The accuracy results of STATPC on data sets where clusters have an equal number of relevant attributes are comparable with the accuracy results of STATPC on data sets where clusters have a different number of relevant attributes.

For most other techniques, the accuracy results on data sets where clusters have an equal number of relevant attributes are slightly higher than the accuracy results on data sets where clusters have a different number of relevant attributes, although the average cluster dimensionality is the same in both cases. This is because, in the latter case, there are implanted clusters with low dimensionality, which are more difficult to retrieve than the implanted clusters with high dimensionality.

Effect of database dimensionality. Figure 6.10 shows the accuracy of

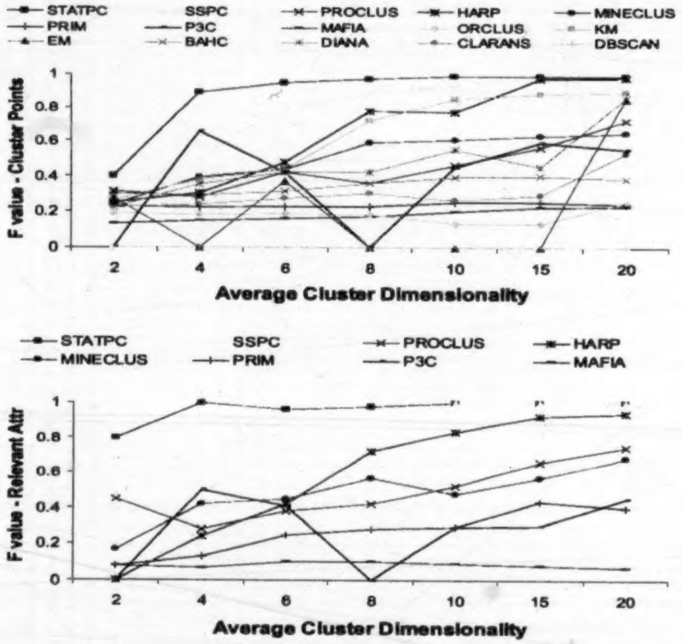

Figure 6.8: STATPC and competing techniques on category Gaussian_Equal

the compared techniques as a function of increasing database dimensionality d. Varying the dimensionality of the data set has a similar effect as varying the average cluster dimensionality: the dimensionality of the clusters varies relatively to the database dimensionality. However, in this experiment we see the effects on a different range of d.

STATPC obtains consistently a high accuracy as the database dimensionality increases, and significantly higher accuracy than the accuracies of the competing techniques.

The accuracies of PROCLUS and ORCLUS decline as d increases, because their initializations in full dimensional space approximate increasingly worse the implanted clusters. Similarly, HARP's accuracy decreases, because HARP favors clusters with many relevant attributes. SSPC's accuracy decreases too, but only slightly, because it becomes increasingly difficult to initialize it with "good" representative points and relevant attributes. MINECLUS accuracy decreases because the parameter β fails to control effectively the trade-off between cluster sizes and number of relevant attributes. The accuracy of PRIM is relatively unaffected by decreasing d, but it has a low value.

The accuracy of MAFIA slightly decreases with increasing d, because MAFIA reports more low dimensional projections of the implanted clusters.

The accuracy of P3C alternates between higher and lower values, depending on how successful P3C is in detecting and aggregating cluster projections.

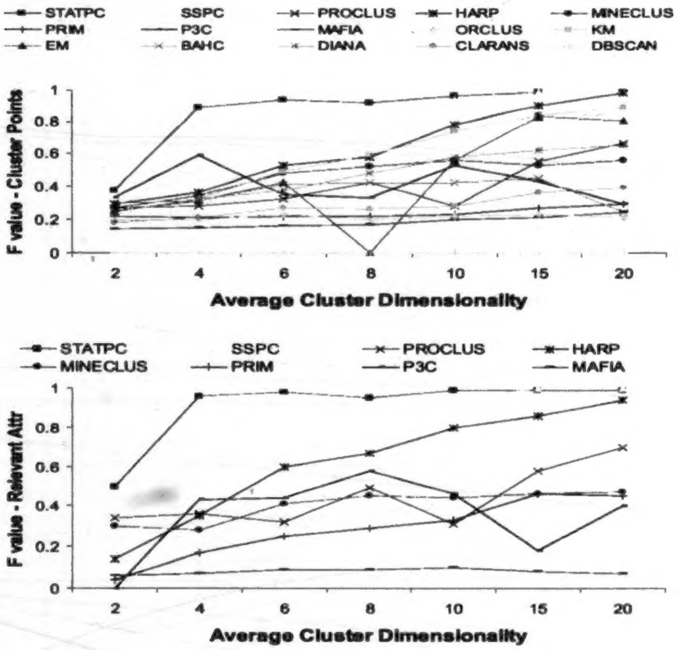

Figure 6.9: STATPC and competing techniques on category Gaussian_Different

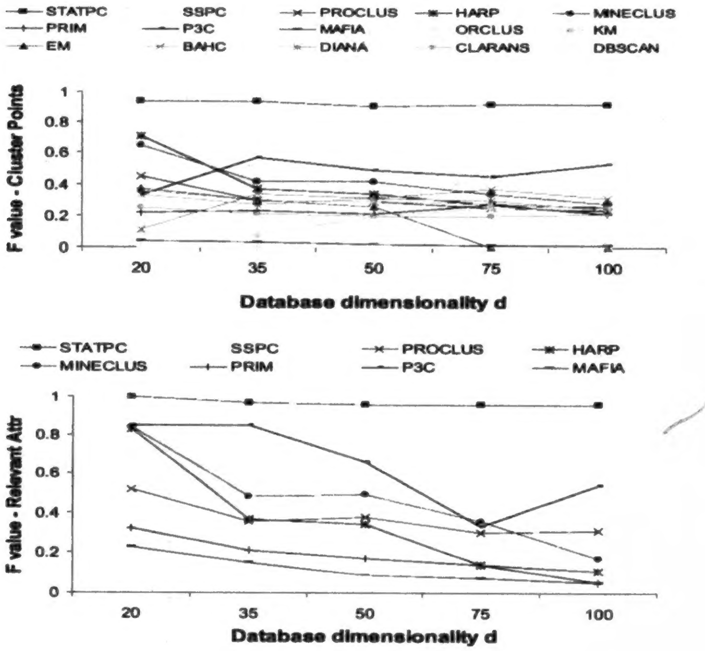

Figure 6.10: The effect of data dimensionality *d* on STATPC and competing techniques

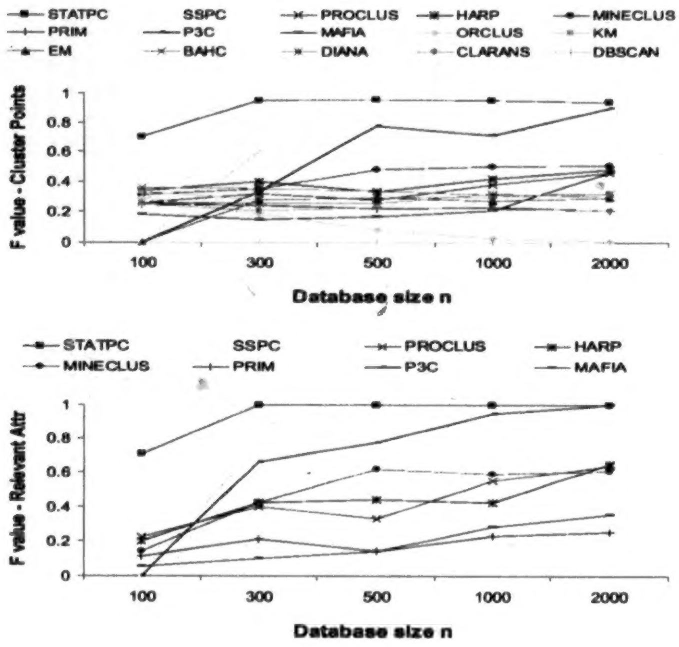

Figure 6.11: The effect of database size n on STATPC and competing techniques

The accuracies of KM, BAHC, DIANA, and CLARANS decrease with increasing d, because the pair-wise distances between data points become more and more similar. The accuracy of EM decreases too, because the quality of the 1-step KM initialization decreases, and because, as d increases, the covariance matrix of each cluster tends to over-fit the data more and more. Again, DBSCAN was unable to detect any clusters.

Effect of database size. Figure 6.11 shows the accuracy of the compared techniques as a function of increasing database size n.

The accuracy of STATPC has a consistently high value once the number of data points is at least 300. When the data set has only 100 data points, STATPC misses the implanted clusters with least points (i.e., the two clusters with 14 points), which are marginally statistically significant.

The accuracies of PROCLUS, HARP, SSPC, MINECLUS, and ORCLUS increase with increasing n, because the more points are in a cluster, the more reliable is the identification of relevant attributes. The accuracy of PRIM is relatively unaffected by increasing n, but it has a low value.

The accuracy of MAFIA also increases with increasing n, because more points in a cluster translate into less $1D$ cluster projections wrongly reported.

P3C's accuracy increases significantly with increasing n, because the $1D$ cluster projections become increasingly detectable, and more evidence is present

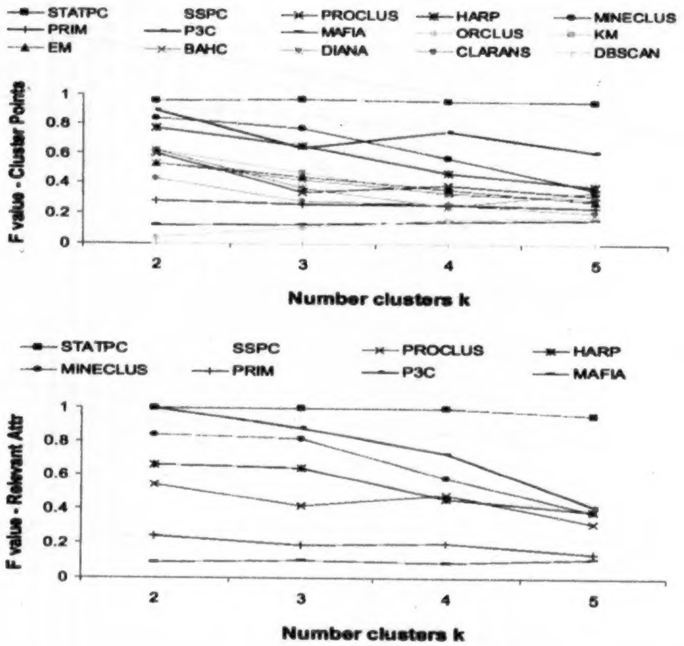

Figure 6.12: The effect of the number of clusters k on STATPC and competing techniques

for their aggregation.

The accuracies of the full dimensional algorithms are relatively unaffected by increasing n. EM's accuracy slightly increases, because more points in a cluster means a more reliable covariance matrix for the cluster. Again, DBSCAN did not find any clusters.

Effect of number of clusters. Figure 6.12 shows the accuracy of the compared techniques as a function of increasing number of clusters k.

STATPC's accuracy remains constantly high as the number of implanted clusters increases.

The accuracies of the majority of techniques decrease as k increases, because larger k means less points per cluster, thus less reliable identification of relevant attributes, less detectable $1D$ cluster projections, and less reliable covariance matrices for clusters. MAFIA and PRIM are relatively unaffected but very inaccurate overall.

Effect of cluster sizes and number of outliers. Figure 6.13 shows the accuracy of the compared techniques as a function of increasing cluster sizes (and consequently, decreasing number of noise points). The points on the x-axis correspond to Setup 1) to 4), as described in Sub-section 6.5.2.

STATPC shows constant, high accuracy as the cluster sizes increase, and significantly higher accuracy than the accuracies of the competing techniques.

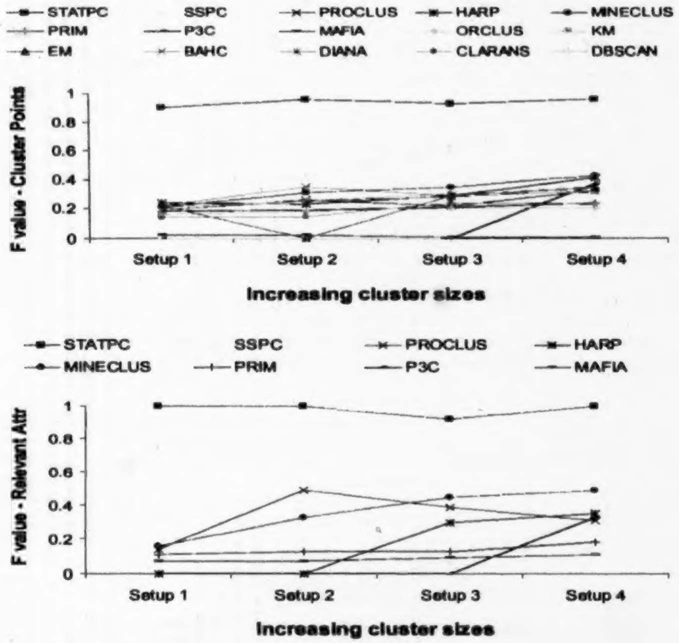

Figure 6.13: The effect of cluster sizes on STATPC and competing techniques

PROCLUS, HARP, SSPC, MINECLUS, and ORCLUS show increased accuracies as the cluster sizes increase, because the identification of relevant attributes is the more reliable, the more points are in clusters.

MAFIA and PRIM are relatively unaffected, but they obtain low accuracy.

Similarly, P3C's accuracy increases, because denser clusters can be more easily detected in individual attributes, and there is more evidence for aggregating the $1D$ cluster projections.

The accuracies of KM, BAHC, DIANA, and CLARANS increase as cluster sizes increase, because these algorithms do not compute outliers, so their accuracies will suffer when the number of outliers is large. EM's accuracy also increases, because the covariance matrices of the clusters are more reliably determined. DBSCAN is unable to detect any clusters.

Effect of extent in relevant attributes. Figure 6.14 shows the accuracy of the compared techniques as a function of increasing extent in relevant attributes. The points on the x-axis correspond to Setup 1) to 4), as described in Sub-section 6.5.2.

The accuracy of STATPC decreases with increasing extent in relevant attributes because the clusters become sparser, and thus less statistically significant.

The accuracies of most competing techniques decrease as the extent of clusters in their relevant attributes increases, because clusters will "stand out"

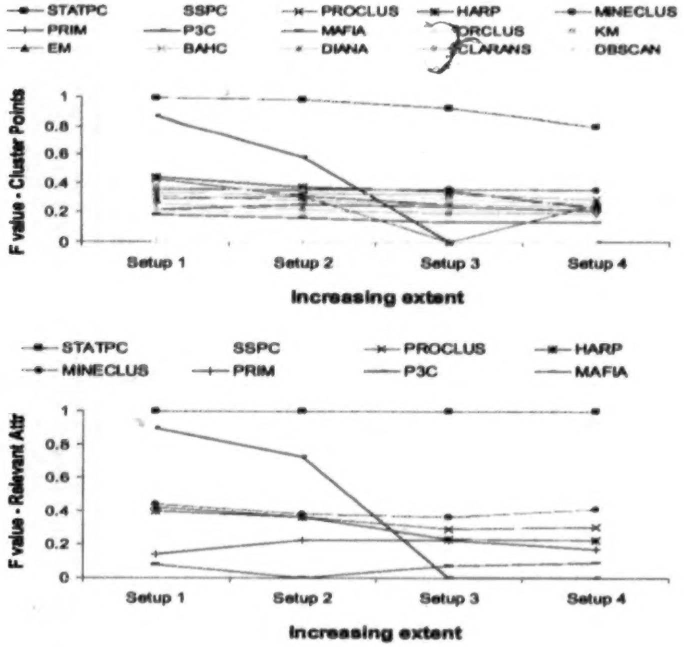

Figure 6.14: The effect of extent of clusters in relevant attributes on STATPC and competing techniques

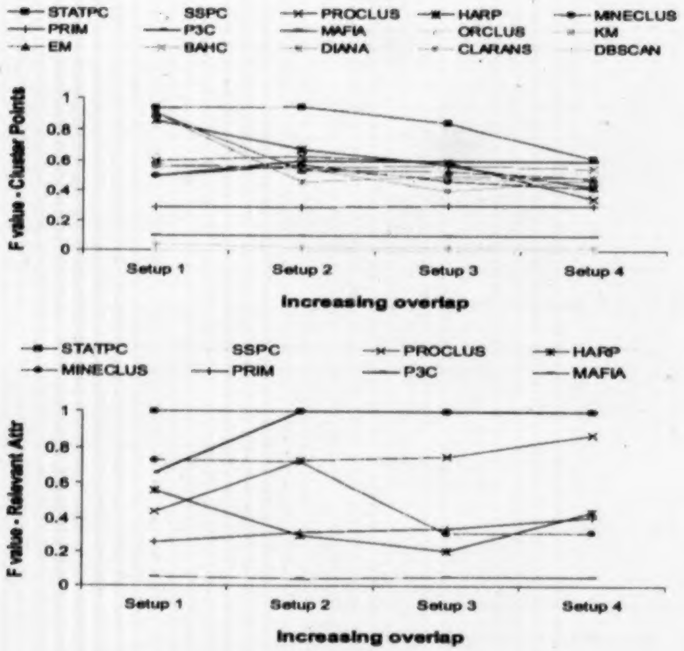

Figure 6.15: The effect of overlap of clusters in common relevant attributes on STATPC and competing techniques

Table 6.1: Analysis of different criteria in data generation

	STATPC	PROCLUS	MINECLUS	HARP	SSPC
Avg. cl. dim. $\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
Unif./Gauss.	$\uparrow_{Unif}$	$\uparrow_{Unif}$	$\uparrow_{Unif}$	$\uparrow_{Unif}$	$\uparrow_{Unif}$
Eq./Diff.	$\sim$	$\uparrow_{Eq}$	$\uparrow_{Eq}$	$\uparrow_{Eq}$	$\uparrow_{Eq}$
Db. dim $\uparrow$	$\sim$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Db. size $\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
Numb. cl. $\uparrow$	$\sim$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Cl. sizes/outl. $\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
Extent $\uparrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Overlap $\uparrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$

less and less in comparison with the uniform background.

Effect of overlap in common relevant attributes. Figure 6.15 shows the accuracy of the compared techniques as a function of increasing overlap between clusters on common relevant attributes. The points on the x-axis correspond to Setup 1) to 4), as described in Sub-section 6.5.2.

The accuracies in terms of cluster points for the majority of the compared techniques, including STATPC, decrease as the overlap between clusters on common relevant attributes increases, because clusters become more and more identical. Some techniques have constant, but poor, accuracy. However, the accuracies in terms of relevant attributes of the compared techniques, including STATPC, increase, because more overlap translates into more points per cluster, and thus, into a more reliable identification of relevant attributes.

Summary of systematic evaluation on synthetic data. We summarize our experimental results in Tables 6.1 and 6.2. The first column contains the effects in the data generation that we have studied, and the first row contains the compared techniques. Arrows indicate increase or decrease in accuracy, and $\sim$ indicates constant accuracy. For instance, the first cell in the Table 6.1 should be read as "the accuracy of STATPC increases as the average cluster dimensionality increases". The notation $\uparrow_{Unif}$ signifies that the accuracy of the technique given in the corresponding column is higher when cluster points are uniformly distributed in their relevant subspace than when the cluster points are Gaussian distributed in their relevant subspace. Similarly, the notation $\uparrow_{Eq}$ means that the accuracy of the technique given in the corresponding column is higher when clusters have an equal number of relevant attributes than when clusters have a different number of relevant attributes.

In general, we observe that the compared techniques show consistent tendencies with respect to the data generation effects studied. All techniques show higher accuracies when cluster points are uniformly distributed in their relevant subspace than when cluster points are Gaussian distributed in their relevant subspace. Also, most techniques obtain higher accuracies when clus-

Table 6.2: Analysis of different criteria in data generation (cont.)

	MAFIA	P3C	PRIM	ORCLUS	Full-dim
Avg. cl. dim. ↑	↑	~	↑	↑	↑
Unif./Gauss.	$\uparrow_{Unif}$	$\uparrow_{Unif}$	$\uparrow_{Unif}$	$\uparrow_{Unif}$	$\uparrow_{Unif}$
Eq./Diff.	$\uparrow_{Eq}$	~	$\uparrow_{Eq}$	$\uparrow_{Eq}$	$\uparrow_{Eq}$
Db. dim ↑	↓	~	~	↓	↓
Db. size ↑	↑	↑	~	↑	~
Numb. cl. ↑	~	↓	~	↓	↓
Cl. sizes/outl. ↑	↑	↑	↑	↑	↑
Extent ↑	↓	↓	↓	↓	↓
Overlap ↑	~	~	↓	↓	↓

ters have an equal number of relevant attributes than when clusters have a different number of relevant attributes. Most techniques have higher accuracies when database size or average cluster dimensionality or cluster sizes increase. Most techniques have lower accuracies when number of clusters or extent of clusters in relevant attributes or overlap of clusters in common relevant attributes increase.

The fact that DBSCAN did not find any clusters in almost all cases reveals a fundamental problem of the density-based paradigm applied on high dimensional data. While these concepts work on data of moderate dimensionality as shown in several publications, the high dimensional data used in our experiments shows a variety of effects of the curse of dimensionality impeding the use of global density-based approaches.

Our results indicate that the denser and/or the more relevant attributes subspace clusters have, the easier it is for the compared techniques to detect these clusters. If clusters are sufficiently dense and/or have enough relevant attributes, even most of the full dimensional clustering algorithms can obtain a good accuracy. We believe that in these cases the selection of the algorithm should be driven by the trade-off between run time and accuracy, and it should be oriented towards techniques with less parameters that, in order to be set, do not require crucial knowledge about the data set (such as the number of clusters or the average cluster dimensionality).

Accuracy results on real data sets. Figures 6.16, 6.17, 6.18, and 6.19 show the accuracy of the compared techniques on the Pima Indians Diabetes, Liver Disorders, WPBC, and Glass data sets and their extensions, respectively, as a function of increased number of uniform attributes added to the data. The first point in the graphs corresponds to the original data sets with no uniform attributes added.

STATPC outperforms the competing techniques on these real data sets, and the largest gap in accuracy between STATPC and the other techniques is obtained on the Pima Indians Diabetes data set.

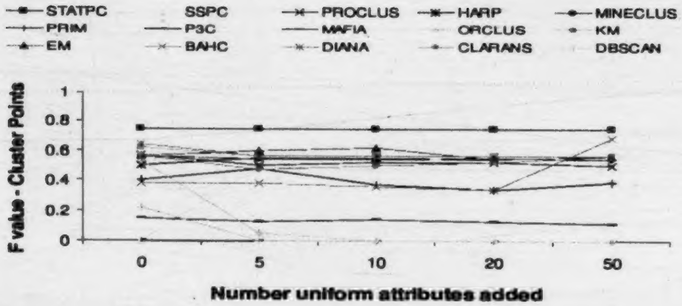

Figure 6.16: Accuracy of STATPC and competing techniques on Pima Indians Diabetes data set

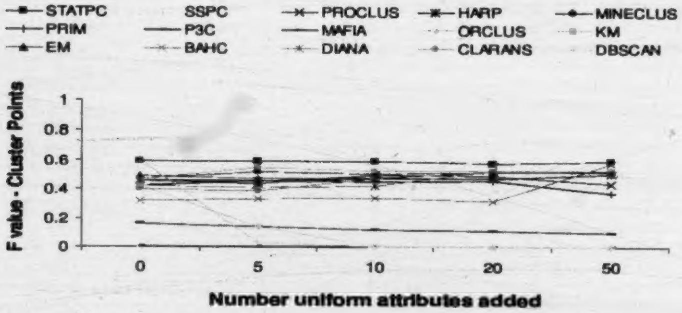

Figure 6.17: Accuracy of STATPC and competing techniques on Liver Disorders data set

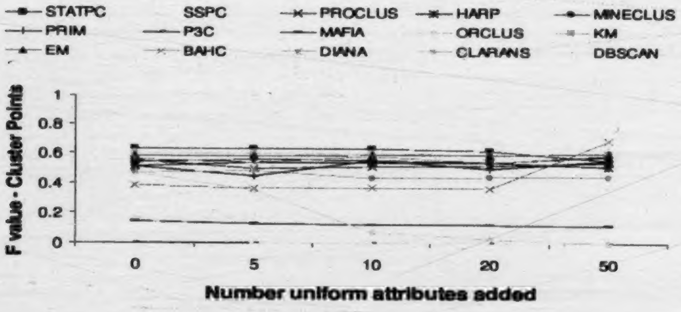

Figure 6.18: Accuracy of STATPC and competing techniques on WPBC data set

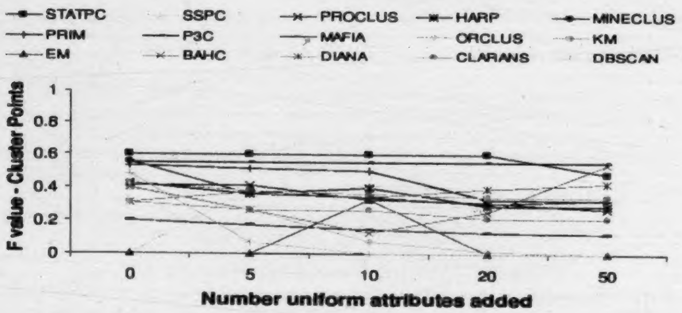

Figure 6.19: Accuracy of STATPC and competing techniques on Glass data set

On the Pima Indians Diabetes data set, STATPC consistently finds 4 8-dimensional clusters. On the Liver Disorders data set, STATPC consistently finds 4 6-dimensional clusters. On the WPBC data set, STATPC finds 2 or 3 33-dimensional clusters. On the Glass data set, STATPC finds 1 or 2 9-dimensional clusters.

The accuracy of a random partition into a number of clusters that equals the number of classes on these data sets is: 0.56 for the Pima Indians Diabetes data set and its extensions; 0.53 for the Liver Disorders data set and its extensions; 0.6 for the WPBC data sets and its extensions; and 0.22 for the Glass data set and its extensions. The accuracy of STATPC on these data sets and their extensions is: 0.75 for the Pima Indians Diabetes data set; 0.59 for the Liver Disorders data set; 0.63 for the WPBC data set; and 0.6 for the Glass data set. In all cases, the accuracy of STATPC is higher than the accuracy of a random partition into a number of clusters that equals the number of classes on these data sets.

The accuracies of most competing techniques decrease as the number of uniform attributes added increases, because it becomes more difficult to detect increasingly lower dimensional clusters. Some of the techniques are not affected by increasing number of uniform attributes, such as P3C and MAFIA.

We have studied the performance of STATPC and the competing techniques on several other real data sets from the UCI machine learning repository, which are comparable in size to the 4 real data sets presented here. In these additional experiments, the accuracy of STATPC is comparable with the best accuracies obtained by the competing techniques, as illustrated in Figure 6.20 for the Iris data set (150 data points, 4 attributes, 3 classes). The accuracy of a random partition into 3 clusters on the Iris data set and its extensions is 0.34. The accuracy of STATPC on the Iris data set and its extensions varies around 0.8.

Accuracy results on larger real data sets. We test STATPC on two gene expression data sets: 1) the colon cancer data set [9] that measures

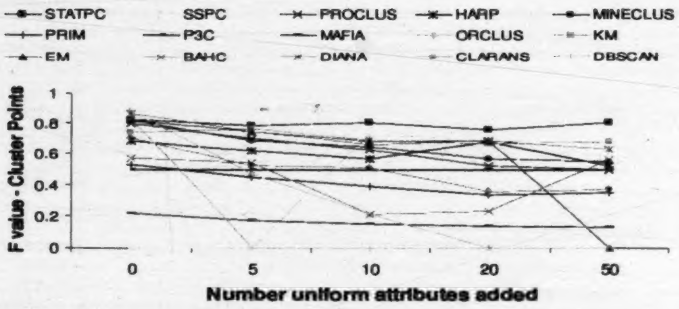

Figure 6.20: Accuracy of STATPC and competing techniques on Iris data set

expression levels of 62 colon tissues (40 tumor and 22 normal colon tissues) in 2000 human genes, and 2) the leukemia data set [37] that measure expression levels of 72 patients (47 patients with acute myeloid leukemia (AML) and 25 patients with acute lymphoblastic leukemia (ALL)) in 7070 human genes. These real data sets are challenging due to the small number of points and the large number of attributes.

On the colon cancer data, STATPC discovers 10 subspace clusters having the following dimensionalities: 1878, 1902, 1973, 1882, 1934, 1752, 1858, 1513, 1900, and 1840. The subspace clusters contain more points from the larger tumor class than from the smaller normal class. The accuracy of STATPC is 0.66. On this data set, MINECLUS, PRIM, and ORCLUS encounter errors in the computation and stop; P3C and MAFIA take unacceptable long time; and EM fails because of singular covariance matrices. Thus, the techniques that can be run on this data set are SSPC, PROCLUS, HARP, KM, BAHC, DIANA, CLARANS, and DBSCAN. DBSCAN does not find any clusters. SSPC computes one 1744-dimensional subspace cluster, and one 1545-dimensional subspace cluster, and its accuracy is 0.45. PROCLUS computes two subspace clusters, one 55-dimensional and one 145-dimensional, and its accuracy is 0.43. HARP computes one 1109-dimensional subspace cluster, and one 1353-dimensional subspace cluster, and its accuracy is 0.52. KM, BAHC, DIANA and CLARANS compute full dimensional clusters, and their accuracies are 0.54, 0.51, 0.51, and 0.45, respectively. The accuracy of a random partition into 2 clusters on this data set is 0.56.

On the leukemia data set, STATPC finds 4 subspace clusters with the dimensionalities: 884, 974, 718 and 618. The subspace clusters contain more points from the larger AML class than from the smaller ALL class. The accuracy of STATPC is 0.74. On this data set, the same techniques as for the colon cancer data set can be run, except SSPC, which encounters errors in the computation and stops. DBSCAN does not find any clusters. PROCLUS computes two subspace clusters, one 10-dimensional and one 190-dimensional, and its accuracy is 0.46. HARP computes one 3935-dimensional subspace

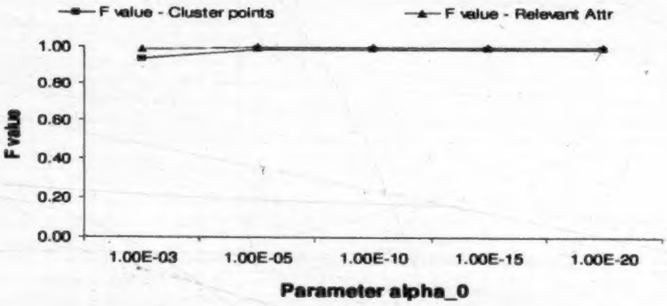

Figure 6.21: Sensitivity of STATPC to parameter α_0

cluster, and one 2094-dimensional subspace cluster, and its accuracy is 0.56. KM, BAHC, DIANA and CLARANS compute full dimensional clusters, and their accuracies are 0.73, 0.42, 0.43, and 0.42, respectively. The accuracy of a random partition into 2 clusters on this data set is 0.56.

We observe that STATPC outperforms the competing techniques both on the colon cancer and the leukemia data sets in terms of accuracy. Also, on these data sets too, the accuracy of STATPC is higher than the accuracy of a random partition into a number of clusters that equals the number of classes on these data sets.

The subspace clusters computed by STATPC are relatively high dimensional, especially on the colon cancer data set. On the leukemia data set, the largest dimensionality of a subspace cluster discovered by STATPC is 13% of the total data dimensionality. The competing techniques, except PROCLUS, also discover clusters with relatively high dimensionality. Relevant attributes of the subspace clusters found represent genes that could be relevant for distinguishing between healthy and cancer tissues, or between different types of cancer. Thus, subspace clusters with less relevant attributes are more useful from this point of view than subspace clusters with many relevant attributes. Note that the full dimensional clustering algorithms do not provide this information at all.

We also note that, except STATPC, all the competing techniques require as critical parameter the target number of clusters, which was set to the number of classes. In addition, none of the techniques, except STATPC, can guarantee the statistical significance of their results.

6.5.8 Sensitivity Analysis

We study the sensitivity of STATPC to its parameters α_0, α_K and α_H. Figures 6.21, 6.22 and 6.23 illustrate the accuracy of STATPC as the parameters α_0, α_K and α_H, respectively, are progressively decreased from $1.0E-3$ to $1.0E-20$ on one of our synthetic data sets.

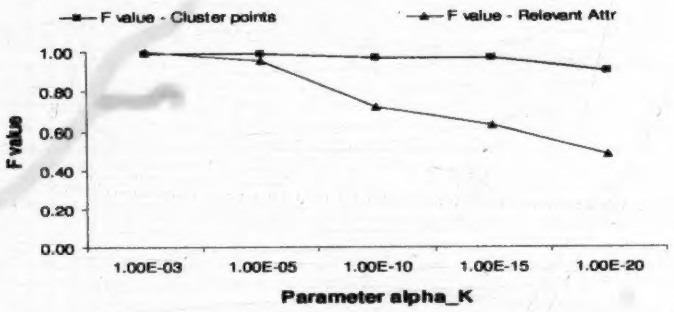

Figure 6.22: Sensitivity of STATPC to parameter α_K

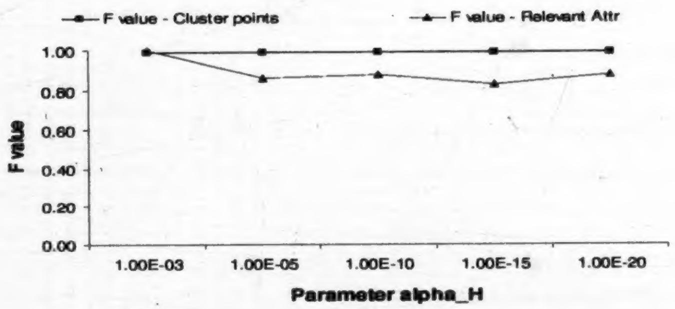

Figure 6.23: Sensitivity of STATPC to parameter α_H

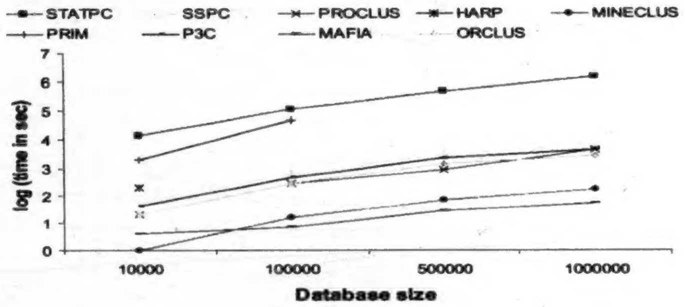

Figure 6.24: Scalability of STATPC and competing techniques with increasing database size

STATPC is relatively robust in terms of the accuracy of the cluster points for all three parameters. In terms of the accuracy of the found relevant attributes, STATPC is robust with respect to parameter α_0, but less robust with respect to parameters α_K and α_H. The effect of parameter α_K is more pronounced because smaller values for this parameter translate into more attributes discarded as irrelevant. Based on this sensitivity analysis, we set $\alpha_0 = 1.0E - 10$, and $\alpha_K = \alpha_H = 0.001$.

6.5.9 Scalability Experiments

In all scalability figures, the time is represented on a log10 scale. We do not need to study the scalability of the full dimensional algorithms because this issue has been studied carefully in the existing literature.

Figure 6.24 shows scalability results for increasing database sizes on synthetic data sets from category $Uniform\_Equal$ with $d = 10$, $K = 2$, 2 relevant attributes per cluster. HARP can be run only on the first data set with $n = 10000$ data points, and PRIM can be run only on the first two data sets with $n = 10000$ and $n = 100000$. Based on their tendencies and the gap between them, the techniques can be partitioned into several groups from smaller to larger running times: MAFIA and MINECLUS in the first group; PROCLUS, ORCLUS, P3C and SSPC in the second group; and STATPC in the last group. The larger runtime of STATPC is due to the construction of numerous MBRs in candidate subspaces around data points Q, and the potentially large number of data points Q tried, but we believe, it is worth the trade-off for much better effectiveness in finding subspace clusters. However, if the scalability of STATPC with the respect to the database size becomes unacceptable for specific applications, one could reduce the number of data points Q tried by choosing these data points Q according to some heuristics, e.g., we could choose data points Q that are far away from subspace clusters already constructed.

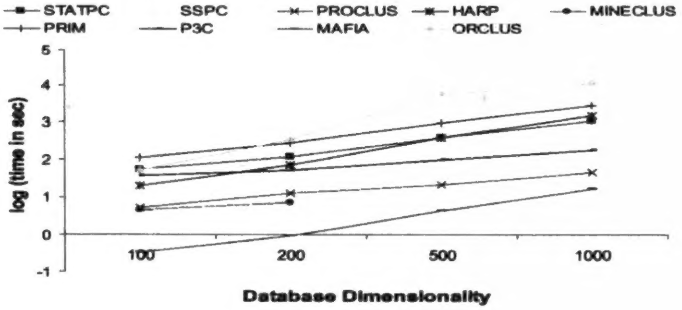

Figure 6.25: Scalability of STATPC and competing techniques with increasing database dimensionality

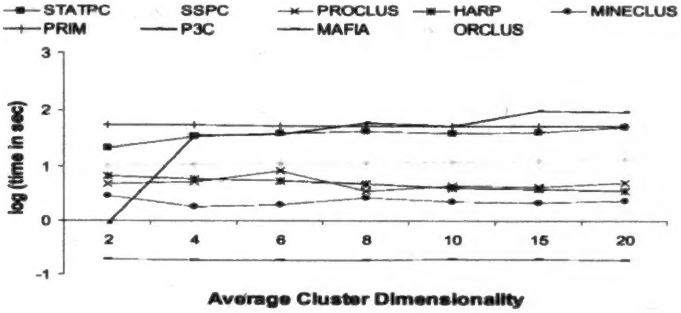

Figure 6.26: Scalability of STATPC and competing techniques with increasing average cluster dimensionality

Figure 6.25 shows scalability results for increasing database dimensionality on synthetic data sets from category $Uniform\_Equal$ with $n = 300$, $K = 2$, 2 relevant attributes per cluster. MINECLUS cannot be run for the last two data sets with $d = 500$ and $d = 1000$. Based on their running times, the techniques can be roughly divided into several groups: MAFIA and SSPC in the first group, followed by PROCLUS in the second group, followed by P3C in the third group, followed by STATPC and HARP in the forth group, followed by PRIM in the fifth group, and finally the group of ORCLUS.

Figure 6.26 shows that the majority of the techniques are unaffected by increasing average cluster dimensionality on data sets from category $Uniform\_Equal$. P3C has exponential complexity in the dimensionality of the largest subspace where clusters exist. MAFIA suffers theoretically from the same problem; however, in this case, MAFIA reports only some $1D$ cluster projections, and thus its run time does not show the expected behavior. STATPC and PRIM form a group of techniques with larger run times than the group of the other techniques, except P3C.

6.6 Summary

In this chapter, we have introduced the algorithm STATPC - an approximation algorithm for the redundancy-aware problem definition given in Section 5.6.

Our extensive experimental evaluation shows that STATPC can effectively discover clusters in the data, even when clusters have low dimensionality with respect to the total dimensionality of the data set. In addition, STATPC significantly and consistently outperforms state-of-the-art projected and subspace clustering techniques, as well as full dimensional clustering algorithms, in terms of accuracy.

We have systematically evaluated and analyzed STATPC, together with numerous, representative projected, subspace and full dimensional clustering algorithms under a variety of experimental conditions on synthetic data sets. We have identified and discussed the strengths and weaknesses of these techniques. We believe that this systematic study can be used by data mining practitioners to decide which techniques are suitable for the problem on hand.

One desirable property of STATPC is that it requires parameters that represent the error probability that the user is willing to accept in statistical tests. These types of parameters are easily understood by users, and can be set without any prior knowledge about the data.

Another desirable property of STATPC is that it can assign a data point to more than one cluster.

The scalability of STATPC with respect to database dimensionality and average cluster dimensionality is comparable to that of the competing techniques. The scalability of STATPC with respect to database size is poorer than that of the competing techniques, but it can be improved through scaling techniques, such as sampling.

The results of STATPC on real data sets are less conclusive than the results of STATPC on synthetic data sets. The results of the compared techniques on real data sets are evaluated using class labels as cluster labels. Although class labels typically indicate some similarities between members of the same class, class labels are not the "perfect" ground truth in the sense that they do not correspond necessarily to subspace clusters in the sense of Definition 5.3. Ideally, the results of a clustering algorithm should be evaluated based on domain knowledge or with the help of a domain expert. But, when the number of algorithms to be evaluated is large, and when many algorithms depend on parameters, such as the number of clusters, whose best values can only be determined through repeated trial-and-error procedures, the evaluation of clustering results becomes tedious. In these situations, class labels provide an acceptable "pseudo"-ground truth that can be used for evaluation.

We underline that STATPC has two important properties that makes it appealing for applying it on real data: 1) we can trust that the solution computed by STATPC stands out in the data in a statistical sense, and it is not just an artefact of the algorithm, and 2) in contrast to most other techniques, STATPC is not based on parameters whose setting requires critical knowledge about the data.

Chapter 7

Conclusions and Future Work

Subspace and projected clustering techniques have been proposed as a solution to the challenges associated with clustering in high dimensional data sets. Subspace and projected clustering techniques are similar in that they detect cluster of points in subsets of attributes, but they differ in their problem definitions, strengths and weaknesses.

Our comprehensive survey of related work in subspace and projected clustering identified several drawbacks of existing work. Subspace clustering techniques are based on global density thresholds, for which no meaningful values is likely to exist; they compute a large number of overlapping clusters; and some of them are sensitive to the resolution of the grid used for density estimation. Projected clustering techniques depend crucially on parameters whose appropriate values are hard to set without critical domain knowledge; they cannot identify well low dimensional clusters embedded in high dimensional spaces; and many of them compute disjoint clusters.

In this thesis, we introduce three novel techniques that advance the state-of-the-art in the subspace and projected clustering field, as discussed below.

First, we propose the projected clustering technique P3C that, in contrast to previously proposed projected clustering techniques: 1) depends on parameters that represent the error probability acceptable by a user in statistical tests, and thus, setting these parameters does not require critical prior knowledge; 2) can effectively discover low dimensional clusters embedded in high dimensional spaces; 3) can compute disjoint or overlapping clusters; 4) is robust to noise in the data.

Second, since there are a few subspace and projected clustering techniques for categorical data, which suffer in general from the same problems as their numerical counterparts, we extend P3C for categorical data. This makes P3C the first projected clustering technique applicable to both numerical and categorical data. P3C for categorical data inherits the properties of P3C for numerical data.

Third, we observe that many subspace and projected clustering techniques have a problem definition that is not independent on the particular algorithm that is proposed for solving the problem. In addition, we cannot assess for the existing algorithms whether the reported clusters are an artefact of the algorithms, or, in fact, these clusters stand out in the data in a statistical sense.

Thus, we propose a novel problem formulation for subspace and projected clustering that aims at extracting from the data non-redundant, axis-parallel regions that stand out in a statistical sense. The problem formulation is given as an optimization problem, which makes our problem formulation independent of a particular algorithm used to solve it.

Since exhaustive search is computationally infeasible, we propose the approximation algorithm STATPC for the optimization problem. STATPC has the following desirable properties: 1) it guarantees that its solution stands out in the data in a statistical sense; 2) its parameters are error probabilities that a user is willing to accept - thus, these parameters are easily set without any domain knowledge; 3) it can discover low dimensional clusters embedded in high dimensional spaces; and 4) it computes overlapping clusters.

In addition, we have evaluated and analyzed STATPC, together with state-of-the-art subspace, projected, and full dimensional clustering algorithms, under a variety of experimental conditions on synthetic data sets. We have discussed strengths and weaknesses of these techniques. We believe that this study not only shows that STATPC outperforms significantly the competing techniques in terms of accuracy, but it can also be used as a guide for the data mining practitioner to select which techniques are preferable in certain scenarios.

7.1 Directions for Future Work

We envision the following interesting directions for future work.

First, from a theoretical point of view, it would be beneficial to have a formal proof for our conjecture that the redundancy-aware problem definition is NP complete.

Second, other approximation algorithms may be proposed for the redundancy-aware problem definition, which may improve upon STATPC in certain aspects, such as scalability or accuracy on real data sets.

Third, one possible explanation for the results of STATPC on real data sets may be that we assume in the *Explain* relationship that all component densities are the uniform density. It would be interesting to study how the *Explain* relationship changes if we change the distribution of the component densities to Gaussian densities, and whether this change improves the results on real data sets.

Forth, in the current problem formulation, we search for axis-parallel regions that have significantly *more* points than expected under uniform distribution. We may as well search for axis-parallel regions that have significantly *less* points than expected under uniform distribution. Such "holes" in the data may be interesting in certain applications.

Fifth, we can also adapt STATPC to categorical data once we are able to compute intervals on categorical attributes. For this, we can use the approach of P3C for computing intervals on categorical attributes.

Bibliography

[1] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek. Detection and visualization of subspace clusters hierarchies. In *DASFAA*, 2007.

[2] E. Achtert, C. Böhm, H. P. Kriegel, P. Kröger, and A. Zimek. Robust, complete and efficient correlation clustering. In *SDM*, 2007.

[3] E. Achtert, H.-P. Kriegel, and A. Zimek. ELKI: a software system for evaluation of subspace clustering algorithms. In *SSDBM*, 2008.

[4] D. Agarwal, A. McGregor, J. Phillips, S. Venkatasubramanian, and Z. Zhu. Spatial scan statistics: approximations and performance study. In *KDD*, 2006.

[5] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD*, 1999.

[6] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD*, 2000.

[7] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, 1998.

[8] R. Agrawal and R. Srikan. Fast algorithms for mining association rules. In *VLDB*, 1994.

[9] U. Alon and al. Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS*, 96:6745–6750, 1999.

[10] M. Ankerst, M. Breuning, H. P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *SIGMOD*, 1999.

[11] I. Assent, R. Krieger, E. Müller, and T. Seidl. DUSC: Dimensionality unbiased subspace clustering. In *ICDM*, 2007.

[12] A. Baddeley. Spatial point processes and their applications. *Lecture Notes in Mathematics*, 1892:1–75, 2007.

[13] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *ICML*, 2002.

[14] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB*, 2002.

[15] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *JRSS-B*, 57:289–200, 1995.

[16] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, 2002.

[17] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? *LNCS*, 1540:217–235, 1999.

[18] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *ICDM*, 2004.

[19] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *SIGMOD*, 2004.

[20] M. M. Breunig, H. P. Kriegel, P. Kröger, and J. Sander. Data Bubbles: quality preserving performance boosting for hierarchical clustering. In *SIGMOD*, 2001.

[21] C. H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *KDD*, 1999.

[22] Y. Cheng and M. Church. Biclustering of expression data. In *ISMB*, 2000.

[23] L. Cheung, K. Y. Yip, D. W. Cheung, B. Kao, and M. K. Ng. On mining micro-array data by order-preserving submatrix. In *International Conference on Data Engineering Workshops*, 2005.

[24] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *SDM*, 2004.

[25] A. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood for incomplete data via the EM algorithm. *J. R. Stat. Soc.*, 39(1):1–38, 1977.

[26] L. Ert öz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *SDM*, 2003.

[27] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *KDD*, 1996.

[28] W. Feller. *An Introduction to Probability Theory and Its Applications.* John Wiley and Sons, NY, 1950.

[29] A. Foss and O. Zaïane. A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. In *ICDM*, 2002.

[30] J. Friedman and N. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9:123–143, 1999.

[31] J. H. Friedman and J. L. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society*, 66(4):815–849, 2004.

[32] G. Gan and J. Wu. Subspace clustering for high dimensional categorical data. *ACM SIGKDD Explorations Newsletter*, 6(2):87–94, 2004.

[33] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS - clustering categorical data using summaries. In *KDD*, 1999.

[34] B. J. Gao, O. L. Griffith, M. Ester, and S. J. M. Jones. Discovering significant opsm subspace clusters in massive gene expression data. In *KDD*, 2006.

[35] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In *VLDB*, 1998.

[36] A. Gionis, A. Hinneburg, S. Papadimitriou, and P. Tsaparas. Dimension induced clustering. In *KDD*, 2005.

[37] T.R. Golub and al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[38] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *SIGMOD*, 1998.

[39] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD*, 1984.

[40] J. Han and M. Kamber. *Data mining: concepts and techniques.* Morgan Kaufmann, 2001.

[41] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, 1998.

[42] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a survey. *ACM Computing Survey*, 31(3):264–323, 1999.

[43] K. Kailing, H. P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *SDM*, 2004.

[44] G. Karypis, E-H. S. Han, and V. Kumar. CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. In *IEEE Computer*, 1999.

[45] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis.* John Wiley and Sons, New York, 1990.

[46] H. P. Kriegel, P. Kröger, M. Renz, and S. Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *ICDM*, 2005.

[47] G. Liu, J. Li, K. Sim, and L. Wong. Distance based subspace clustering with flexible dimension partitioning. In *ICDE*, 2007.

[48] J. Liu and W. Wang. OP-Cluster: Clustering by tendency in high dimensional space. In *ICDM*, 2003.

[49] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematics, Statistics and Probability*, 1967.

[50] S. C. Madeira and A. J. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE TCBB*, 1(1):24–45, 2004.

[51] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *KDD*, 2008.

[52] G. Moise, J. Sander, and M. Ester. P3C: A robust projected clustering algorithm. In *ICDM*, 2006.

[53] G. Moise, J. Sander, and M. Ester. Robust projected clustering. *Knowledge and Information Systems*, 14(3):273–298, 2008.

[54] H. Nagesh, S. Goil, and A. Choudhary. Adaptive grids for clustering massive data sets. In *SDM*, 2001.

[55] K.K.E. Ng, A.W. Fu, and C.-W. Wong. Projective clustering by histograms. *IEEE TKDE*, 17(3):369–383, 2005.

[56] R. T. Ng and J Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, 1994.

[57] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations Newsletter*, 6(1):90–105, 2004.

[58] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. Maple: a fast algorithm for maximal pattern-based clustering. In *ICDM*, 2003.

[59] A. Pothen, H. D. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11(3):430–452, 1990.

[60] PRIM package. `http://cran.r-project.org/web/packages/prim/`.

[61] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T.M. Murali. A Monte Carlo algorithm for fast projective clustering. In *SIGMOD*, 2002.

[62] R package. `http://www.r-project.org/`.

[63] K. Sequeira and M. Zaki. SCHISM: a new approach for interesting subspace mining. In *ICDM*, 2004.

[64] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB*, 1998.

[65] G. W. Snedecor and W. G. Cochran. *Statistical Methods*. Iowa State University Press, 1989.

[66] A. K. H. Tung, X. Xu, and B. C. Ooi. CURLER: finding and visualizing nonlinear correlation clusters. In *SIGMOD*, 2005.

[67] UCI Machine Learning Repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[68] H. Wang, F. Chu, W. Fan, P. S. Yu, and J. Pei. A fast algorithm for subspace clustering by pattern similarity. In *SSDBM*, 2004.

[69] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *SIGMOD*, 2002.

[70] W. Wang, J. Yang, and R. R. Muntz. STING: A statistical information grid approach to spatial data mining. In *VLDB*, 1997.

[71] X. Wang and H. J. Hamilton. DBRS:a density-based spatial clustering method with random sampling. In *PAKDD*, 2003.

[72] X. Wang and H. J. Hamilton. Density-based spatial clustering in the presence of obstacles and facilitators. In *PKDD*, 2004.

[73] X. Wang and H. J. Hamilton. Towards an ontology-based spatial clustering framework. In *Eighteenth Canadian Artificial Intelligence Conference*, 2005.

[74] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee. FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[75] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[76] X. Xu, Y. Lu, A. K. H. Tung, and W. Wang. Mining shifting-and-scaling co-regulation patterns on gene expression profiles. In *ICDE*, 2006.

[77] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: capturing subspace correlation in a large data set. In *ICDE*, 2002.

[78] K. Y. Yip, P. Qi, M. Schultz, D. W. Cheung, and K. H. Cheung. SemBiosphere: a semantic web approach to recommending microarray clustering services. In *PSB*, 2006.

[79] K.Y. Yip, D.W. Cheung, and M.K. Ng. HARP: a practical projected clustering algorithm. *IEEE TKDE*, 16(11):1387–1397, 2004.

[80] K.Y. Yip, D.W. Cheung, and M.K. Ng. On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In *ICDE*, 2005.

[81] M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In *ICDM*, 2003.

[82] M. L. Yiu and N. Mamoulis. Iterative projected clustering by subspace mining. *IEEE TKDE*, 17(2):176–189, 2005.

[83] O. Zaïane and C. H. Lee. Clustering spatial data when facing physical constraints. In *ICDM*, 2002.

[84] M. Zaki, M. Peters, I. Assent, and T. Seidl. CLICKS: an effective algorithm for mining subspace clusters in categorical datasets. In *KDD*, 2005.

[85] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *SIGMOD*, 1996.

[86] Y. Zhang, A. W. Fu, C. H. Cai, and P. A. Heng. Clustering categorical data. In *ICDE*, 2000.

[87] L. Zhao and M. J. Zaki. Microcluster: Efficient deterministic biclustering of microarray data. *IEEE Intelligent Systems*, 20(6):40–49, 2005.

Appendix A

Statistical Hypothesis Testing

Statistical hypothesis testing is a formal way of choosing between two competing claims/hypotheses: the *null* hypothesis, denoted by H_0, and the *alternative* hypothesis, denoted by H_a. The null hypothesis represents a theory that has been put forward. Special consideration is given to the null hypothesis because it relates to the statement being tested, whereas the alternative hypothesis relates to the statement to be accepted when the null hypothesis is rejected.

In order to set up a statistical hypothesis test, the null hypothesis H_0 must be formulated, and a *test statistic* must be chosen. The test statistic is a number that summarizes the information in the data that is relevant for H_0. Then, the distribution of the test statistic under H_0 is determined. The result of a statistical hypothesis test is given in terms of the null hypothesis. It is either "Reject H_0" or "Do not reject H_0".

The *significance level* α of a statistical hypothesis test is a fixed probability of wrongly rejecting the null hypothesis, when in fact it is true. α is also called the rate of false positives or the probability of type I error.

The *critical value* of a statistical hypothesis test is a threshold to which the value of the test statistic is compared to determine whether or not the null hypothesis is rejected. For a one-sided test, the critical value θ_α is computed based on the equation

$$\alpha = Probability(Test\_statistic >= \theta_\alpha) \tag{A.1}$$

and for a two-sided test, the right critical value θ_α^R is computed by (A.1), and the left critical value θ_α^L is computed based on the equation

$$\alpha = Probability(Test\_statistic <= \theta_\alpha^L) \tag{A.2}$$

where the probability is computed in each case using the distribution of the test statistic under the null hypothesis.

Figure A.1 illustrates a distribution of a test statistic under a null hypothesis. θ_α^R and θ_α^L are the right, respectively left, critical values of the statistical test. The significance level α equals the area under the curve to the right of θ_α^R, and it also equals the area under the curve to the left of θ_α^L. When

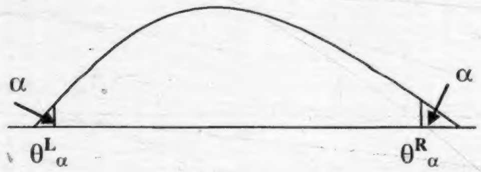

Figure A.1: Two-sided statistical hypothesis test

the test statistic is greater than θ_α^R or smaller than θ_α^L, the null hypothesis is rejected. Equivalently, the probability that the test statistic has its current value, provided that the null hypothesis is true, is very small, i.e., it is less than α.

The probability of not rejecting the null hypothesis, when in fact it is false, is called the rate of false negatives or the probability of type II error, and it is denoted by β. A type I error is often considered more serious; thus α is set to a suitable small value. The exact probability of a type II error is generally unknown, because the distribution of the test statistic under H_a cannot be generally determined. Type I and type II errors are inversely related; the smaller the risk of one, the higher the risk of the other.